DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Rich Linguistic Structure from Large-Scale Web Data

Citation

Yamangil, Elif. 2013. Rich Linguistic Structure from Large-Scale Web Data. Doctoral dissertation, Harvard University.

Permanent link

http://nrs.harvard.edu/urn-3:HUL.InstRepos:11181110

Terms of Use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Rich Linguistic Structure from Large-Scale Web Data

A dissertation presented
by

Elif Yamangil

to
The School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
July 2013

©2013 - Elif Yamangil

All rights reserved.

Dissertation advisor

Professor Stuart M. Shieber

Author

Elif Yamangil

Rich Linguistic Structure from Large-Scale Web Data

Abstract

The past two decades have shown an unexpected effectiveness of *Web-scale* data in natural language processing. Even the simplest models, when paired with unprecedented amounts of unstructured and unlabeled Web data, have been shown to outperform sophisticated ones. It has been argued that the effectiveness of Web-scale data has undermined the necessity of sophisticated modeling or laborious data set curation. In this thesis, we argue for and illustrate an alternative view, that Web-scale data not only serves to improve the performance of simple models, but also can allow the use of qualitatively more sophisticated models that would not be deployable otherwise, leading to even further performance gains.

We investigate this hypothesis through the use of both *parametric* and *Bayesian nonparametric* modeling techniques. First, by comparing rich parametric models against simpler models, we show that richer modeling of Web data brings about qualitative and quantitative performance gains. Experimental results in the application domain of *sentence compression* show that richer models lead to systems with improved robustness and generalization power. In the domain of *lexical correction* we augment a coarse *generative model* with a *discriminative reranking* component that incorporates richer contextual information

and achieve improvements in performance.

Second, by using the Bayesian nonparametric modeling framework, we show how to induce rich models in fully automatic, data-driven ways as opposed to heuristically. We propose principled and unified solutions to not only the estimation but also the *model selection* problem of the linguistically sophisticated grammar formalisms of *tree-insertion grammars* and *tree-adjoining grammars*. When evaluated in the domain of *syntactic parsing*, our induced grammars do not only lead to improved performance but are also *compact*, allowing for efficient computational processing and linguistic analysis. Further we show that these compact grammars can achieve the same performance gains as the parametric rich models in Web-scale sentence compression experiments.

Contents

List of Figures

List of Tables

Acknowledgments

I am hugely indebted to my advisor, Stuart Shieber. I received excellent feedback from him whenever I asked for it but at the same time was given tremendous confidence and intellectual space to grow on my own. The frequency of his "big-picture moments" and the clarity with which he was able to explain his insights was astounding.

I owe a significant portion of the inspiration for this thesis to Rani Nelken, who was a post-doc at the time I joined the lab. His vision for the effectiveness of dynamic Web was clearly ahead of its time. It makes me proud to see our first papers mentioned as examples of the early work using Wikipedia revisions.

The AIRG community — Barbara Grosz, Radhika Nagpal, David Parkes among others — provided a welcoming environment, especially for a female foreign student who felt like an absolute fish out of the water. My floor-mates at Conant Hall — Vasily Dzyabura, Behtash Babadi, Lami Kim, Yuka Minagawa, François-Xavier Willems, Alejandro Perdomo and our resident advisor Yanyan Liu — and my friends at Harvard and Brown — Ece Kamar, Ibrahim Eden, Alptekin Kupcu, Gulen Oktar, Gavril Bilev — made those very lonely first few years in a strange country fun and friendly.

I spent most of my third year at the Harvard statistics department. I am indebted to professors Samuel Kou, Edoardo Airoldi, Patrick Wolfe, Joseph Blitzstein, and their outstanding teaching staff and students for many helpful discussions. A special thank you to Jung Ook Hong who endured those painful STAT 221 problem sets with me.

I have had the opportunity of working directly with several professors. Professor Peter Bol and his team at the Institute for Quantitative Social Science provided me with a project that not only funded my studies but also gave me a set of very intriguing problems to work on in the domain of Chinese history. Professors Barbara Grosz and Ryan Adams provided

helpful discussions and feedback on various stages of the dissertation manuscript.

I could only dream of being so lucky to have someone like Heather Pon-Barry as my officemate. I relied on Heather countless times for help with my broken English, my social and cultural ineptitude during the first few years, my incompetence in keeping up with Harvard and all of its bureaucracy, and so much more. She was a sister and a role model to me. I would also like to acknowledge the friendship of Aysegul Topcu, Anil Usumezbas, Ali Osman Ulusoy, Ozge Can Ozcanli, Murat Mungan, the Knitting Ladies, the Pink House and its extended community.

Thanks to my parents Behiye and Fuat Yamangil and my brother Emre, for their love and support throughout these years. They got me through the rock bottom of my PhD when I suddenly took a flight back home and even became physically ill. Nevertheless, nothing could shake their confidence in me, which provided the solid foundation I would rest on to get through each day. Also, thanks to Paul and Carl Swanson and Ben's godparents Michele and Tom Wolfson for providing the warmth and tradition of an American family.

Lastly and most importantly, I would like to thank Ben Swanson, for finding me, and then finding me again; for being the one person who cared for me enough to break through the walls of workaholism that I had put up. I now understand that most successful people have the privilege of a loving life partner who believes in them perhaps even more than they believe in themselves. Thank you, Ben, without your support and example, and not to forget the friendly competition (!) that you bring into our lives, I would not have been nearly as satisfied with this work as I am now.

This thesis work was supported in part by a Google PhD fellowship in natural language processing.

For my family Behiye, Fuat, Emre Yamangil,
and Ben Swanson.

Chapter 1

Introduction

A large and increasing proportion of human discourse now occurs on the Internet — on social networking sites, online commerce sites, video-sharing sites, news networks, blogs, wikis, and over email. Halevy et al. (2009), emphasizing the explosive growth of natural language samples on the Web, have argued that **"Simple models and a lot of data trump more elaborate models based on less data."** In other words, as long as there is more raw data available on the Web, "in the wild", one can throw more of it at any *simple* model and achieve superior results without having to worry about creating manually annotated data sets or crafting *sophisticated* models. In this thesis, we[1] outline work that will help us argue a consistent but complementary view: *Web-scale* data not only serves to improve the performance of simple models, but also can help to afford the use of qualitatively more sophisticated models that would not be deployable otherwise.

[1] Since some of the chapters of the thesis are based on work that was done collaboratively, as specified in footnotes in individual chapters, the first person plural pronoun will be used throughout the thesis, even when the contribution is due solely to the author of the thesis.

Simpler models have a natural advantage over sophisticated models, especially in the statistical, data-driven framework: roughly speaking, as the linguistic sophistication of the statistical models increase, so does the severity of the scarcity of annotated data samples. More parameters means more need for data samples that carry the particular signals pertaining to model parameters, which typically means more annotated data. Annotating data manually is costly and time consuming; therefore researchers have had to resort to unsupervised models or utilizing indirect signals, at the cost of significant decrease in predictive accuracy.

In this thesis, we show that the Web, through its dynamic evolution as it is maintained and developed by its users across the globe, can be a significant source of exactly the kind of annotation that is needed for certain natural language processing (NLP) tasks. We show that Web-scale annotated data is valuable because (1) it permits the deployment of a *parametric* model of finer granularity, and (2) in combination with the *Bayesian nonparametric* modeling framework, it can serve to induce models of finer granularity in a fully data-driven way. We demonstrate (1) by hand-crafting finer-grained models in particular tasks and obtain performance gains over coarser model alternatives. In support of (2), we design a portfolio of Bayesian nonparametric inference techniques for grammar formalisms with ever richer linguistic structure. We show how these techniques answer the question of model fit, fight data-specific over- or under-fitting issues, and converge on rich representations that achieve even further performance boosts, especially in combination with the large-scale annotated data samples that we collect from the Web.

In detail, we start by discussing our experiments with data mining of Wikipedia article revision histories for natural language samples; we describe how we obtain orders of

magnitude more data for a number of tasks than can be obtained by traditional means such as manual annotation. In Chapter 3, we present fine-grained, sophisticated models that are permitted by the availability of large-scale data, and show their improved performance against simpler models within the application domains of extractive sentence compression and lexical correction. In Chapter 4, we explore the promise of the Bayesian nonparametric framework to determine the optimal level of sophistication of our models in a fully data-driven way. We design *Markov chain Monte Carlo (MCMC)* inference techniques that make feasible solving the difficult search problem of finding the right level of sophistication under increasingly expressive grammar formalisms that lie between tree-substitution grammars and tree-adjoining grammars. The inference framework that we build enables us to explore this space in a principled way using the task of syntactic parsing as our testbed. Therefore we carry out a comparative analysis between grammar formalisms with varying levels of linguistic expressivity in terms of predictive accuracy, compactness of representation, and computational efficiency. In Chapter 5, we demonstrate the power of our inference framework on Web data sentence compression experiments.

1.1 Web 2.0: collecting parallel data

It is not surprising that, paralleling the accumulation of an immense amount of natural language data on the Web, the trending methodology in NLP has been statistical. For example, beginning with the IBM machine translation (MT) models (Brown et al., 1990), a plethora of other statistical approaches have been applied to MT. The basic word-level approach gave way to phrase-based (Koehn et al., 2003; Och and Ney, 2004), hierarchical phrase-based (Wu, 1997; Chiang, 2005), and syntax-based approaches (Yamada and

Knight, 2001; Gildea, 2003), gradually increasing in the granularity of linguistic structure being used. Similarly, as an interesting application of the advances in the statistical MT (SMT) methodology, tasks of monolingual translation — examples of which include paraphrasing, sentence compression, text correction, and question answering — have received considerable attention from researchers in statistical NLP (Knight and Marcu, 2002; Quirk et al., 2004; Cohn and Lapata, 2008; Wang et al., 2007).

However, roughly speaking, as the linguistic sophistication of the statistical models increase, so does the severity of the scarcity of cleanly aligned parallel data (for example, sentence pairs that are translations of each other), creating a bottleneck especially for tasks of monolingual translation. In research on paraphrasing, researchers have had to resort to using very small data sets that are manually created (Cohn and Lapata, 2008), or "comparable data" collected from the Web (sentence pairs that are not necessarily paraphrases of each other, but are semantically related, for example sentences covering the same news story) (Barzilay and Lee, 2003), and some have gone so far as to specifically tailor their models to capture monolingual phenomena from bilingual data, which is more abundantly available from the SMT community (Bannard and Callison-Burch, 2005). The first of these approaches is not a step towards the vision that computers would learn from immense Web data, and is clearly not scalable. The other two approaches suffer from noise due to being indirect. Unfortunately, it is difficult to utilize fine-grained syntax-based models without large amounts of parallel data.

The past decade has witnessed the adoption of the view that the Internet is becoming a "platform". Users no longer passively view the static content that is the Web, created by a few users on a desktop somewhere, but they themselves actively participate in collectively

creating the content that they view. Examples of these platforms are pervasive — social networking sites, blogs, wikis, video-sharing sites — all sharing the common denominator of having user-generated content (UGC). This change in the nature of the Web, commonly referred to as "Web 2.0", means that the Web is evolving from its initial static form of being a globally accessible collection of documents, into a kind of living organism, dynamic and self-sustainable.

Interestingly, much of these dynamics are archived. The Wayback Machine[2] allows users to browse through billions of Web pages archived from 1996 to a few months ago. Search engines index and provide instant access to Web pages created over time. Wikipedia archives and facilitates browsing of the revisions of all its articles. News stories appear on news networks as well as personal Web pages such as blogs and social-networking sites. The stories get copied over, spreading throughout the Web, often resulting in quotes and phrases from these stories to quickly form informational cascades and turn into "memes". The newfound accessibility of the time dimension affects the ways in which we observe and understand our world. Artificial intelligence researchers have started studying the dynamic Web phenomena: news cycles, the blogosphere, social network structure, instant messaging networks (Leskovec et al., 2009; Goetz et al., 2009; Leskovec and Horvitz, 2008). The NLP community also has enlisted the Web as a general purpose corpus of text (Lapata and Keller, 2005; Whitelaw et al., 2009; Quirk et al., 2004) as well as modeling the topic/content dynamics of collections of documents across time and across social networks (Hall et al., 2008; McCallum et al., 2007; Chang et al., 2009).

We see Wikipedia as an environment well suited to the study of human text-editing

[2] `http://www.archive.org/web/web.php`

behavior and language change in UGC-based Web platforms. In order to illustrate, we provide examples of human text-editing behavior found in Wikipedia revisions. We limit ourselves to sentence edits: revisions at the sentence-level only, but see our prior work for a broader investigation from the word to the sentence to the document-level dynamics of article revisions (Nelken and Yamangil, 2008). In the following examples, the first sentence is always the version prior to the edit (b: for before), the sentence that follows is the edited form (a: for after), and we emphasize the change being made using boldface.

1. Grammar/stylistic correction

    b: The economies of both sides have been hurt by their inability to make substantial progress toward a peaceful resolution and mutual economic blockades.

    a: The economies of both sides have been hurt by their inability to make substantial progress toward a peaceful resolution and **by** mutual economic blockades.

2. Extractive compression

    b: The economies of both sides have been hurt by their inability to make substantial progress toward a peaceful resolution **and by mutual economic blockades**.

    a: The economies of both sides have been hurt by their inability to make substantial progress toward a peaceful resolution.

3. Abstractive compression

    b: The economies of both sides have been hurt by their inability to **make substantial progress toward** a peaceful resolution.

    a: The economies of both sides have been hurt by their inability to **achieve** a peaceful resolution.

4. Paraphrasing

    b: The economies of both **sides** have been hurt **by their inability to achieve** a peaceful resolution.

a: The economies of both **countries** have been hurt **in the absence of** a peaceful resolution.

We have collected on the order of millions of such examples[3] from article revisions using simple edit distance between the adjacent revisions. There are a number of desirable properties of this data set: It is not only free and abundant, but it also is by its very nature aligned at the sentence-level and most of the time at the phrase-level as well, reducing the complexity of the inference task, which we will describe below.

1.2 Applications

In order to demonstrate how the abundance of annotated data can in fact lead to performance gains through the usage of sophisticated models, we turn to NLP application domains, namely sentence compression and lexical correction (Chapter 3). In our extractive sentence compression experiments, we use a subset of this data, namely any sentence that was compressed by an editor by dropping words and keeping the word order intact, to train and test a highly specialized synchronous grammar, in particular, a lexicalized *synchronous context-free grammar (SCFG)*. Due to the drastically increased number of parameters in this model, training was made possible by the availability of Web-scale data (over 380,000 sentence pairs, 2 orders of magnitude more than the number of pairs in the standard corpus for this task: about a thousand sentence pairs). In our comparison against the state-of-the-art unlexicalized SCFG, the lexicalized SCFG performed comparably even when it was tested out of domain, which demonstrates the improved generalization power

[3]Examples that do not fall into any of these categories are mostly changes made to the informational content of the sentences.

of our overall approach.

Next, we propose a novel model of large-scale lexical correction of all document words, including both context-sensitive spelling correction and stylistic lexical modifications. In this task, we wish to correct all possible textual errors, rather than focusing on a set of predetermined target words, making the learning problem much more difficult. Since Wikipedia articles are edited collaboratively, errors introduced by one writer are likely to be subsequently corrected by others. We mine a set of 1.5 million such correction training samples. We use the Wikipedia data to train a novel model of text correction, based on a *generative hidden Markov model (HMM)* and a *discriminative reranking perceptron* that incorporates fine-grained contextual information, forming an effective model of correction. We evaluate the richer combination model against the simpler and coarser HMM-only model in the domain of context-sensitive spelling correction and achieve better performance with the addition of the fine-grained information based on the availability of the Web-scale data.

1.3 Bayesian models of syntax

So far, we highlighted the availability of abundant parallel data allowing the use of qualitatively more sophisticated models. Underlying this problem is a *statistical grammar induction (SGI)* problem. Does the availability of Web-scale data allow for the induction of grammars based on qualitatively more sophisticated and expressive grammar formalisms? A positive answer to this question would allow leveraging of more data to improve performance both because of the data itself and because of the enabling of better models. To answer this question, we must address issues of model choice, including the grammar

formalism and inference scheme.

Available grammar formalisms themselves vary in linguistic sophistication and expressivity. For example, flat and sequential *finite-state formalisms* such as n-grams and hidden Markov models are powerful lexical memorizers but they fail to capture the embedded recursive nature of the data. On the other hand, *context-free grammars (CFG)*, despite their ease of implementation and estimation, are known to be inadequate in modeling the syntacticolexical dependencies in force, such as the lexical information that makes n-gram models so effective (Magerman, 1995; Charniak, 1997; Collins, 2003). An alternative way to model languages is to use *tree-substitution grammars (TSG)*, which replace the single-level parent-and-child elementary trees of CFG with a set of multi-level trees, thereby relaxing the independence assumptions of CFG, or *tree-adjoining grammars (TAG)*, which allows for the splicing in of material in the middle of a TSG-like elementary tree, thereby capturing more *reuse* of elementary trees in training data. (See Chapter 2 for a detailed discussion of various grammar formalisms.)

Though formalisms with sophisticated combinatory operations may be better tuned to the dependencies in natural language, their utility depends on sufficiently effective inference algorithms. The conventional view of inference consists of parametric generative models that are estimated via *supervised* training (where closed-form solutions are available) or *unsupervised* training (where a scalable general-purpose algorithm, typically *expectation maximization (EM)* (Dempster et al., 1977) applies). However, by assuming a fixed parametric model space and involving no mechanism for encoding prior domain knowledge about this space, this view is subject to the dual problems of *overfitting* and *underfitting*. Roughly speaking, these are the problems of using a model with too many

and too few *degrees of freedom* with respect to a specific data set, respectively.

Recent work in SGI has therefore addressed these two problems by focusing on *model selection*: What level of model complexity is "right"? This work has led to regularization methods such as hierarchical mixture models (Blei et al., 2003), *Dirichlet priors* for *multinomially* distributed parameters commonly found in NLP (Johnson, 2007; Goldwater and Griffiths, 2007), nonparametric models and stochastic process priors such as the *Dirichlet process (DP)* (Teh et al., 2006; Liang et al.; Van Gael et al., 2009),[4] as well as effective algorithms such as MCMC methods, *variational approximate inference*, and *expectation propagation* (Bishop, 2007). These methods prevent overfitting by introducing a controllable trade-off between the data and prior expectations about the appropriate model complexity and structure, as well as better accounting for the uncertainties that result from having observed a finite amount of data (Goldwater and Griffiths, 2007). On the other hand, nonparametric priors allow for the number of parameters (a natural proxy for model complexity) to grow with the data whenever appropriate, therefore preventing underfitting.

Given these advances in Bayesian nonparametric formulations, now the question of effective SGI boils down to the question of finding a grammar formalism expressive enough to capture the desired patterns in the data, and at the same time amenable to efficient computational processing. The problem with sophisticated formalisms is the very complexity of model selection. As we will see in the following chapters, with TSG one must reason about exponentially many derivations that may underlie the observations (see Figure 2.1). With TAG, the problem only becomes more daunting as one must reason about not only

[4] In this thesis we will mostly focus on Bayesian regularization methods since they fit naturally within the generative probabilistic framework that we will subscribe to.

the substitution operation but also auxiliary trees and their adjunctions (see Figure 2.2).

Interestingly, recent work in parsing has utilized the model selection capabilities of the Bayesian nonparametric framework to bring an elegant solution to TSG induction (Cohn et al., 2009; Post and Gildea, 2009). An independent collection of DP priors and a Gibbs sampling algorithm are used to explore the space of TSGs underlying an entire treebank corpus, without making any hard assumptions about the number and identity of the elementary trees in the grammar. Instead, the frequencies of patterns in the data efficiently determine what these should be. Despite the lack of intensive knowledge engineering going into the system, evaluations have shown performance levels competitive with the heavily-tuned state-of-the-art systems.

The Bayesian nonparametric framework holds promise as a principled way of determining the appropriate parametrization (level of granularity) of induced grammars within a family of grammars defined by a grammar formalism. However, the choice of grammar formalism to begin with remains an open question. We address this question by exploring the space of grammar formalisms (including TSG, TIG[5] and TAG), doing nonparametric induction within each grammar formalism, and analyzing performance using the test-bed task of syntactic parsing[6]. In Chapter 4, we design and implement novel representations and MCMC sampling algorithms for grammar formalisms from least (TSG) to most (TAG) linguistic expressivity. We make sure our sampling techniques are effective and efficient so that they give us correct solutions with a reasonable amount of computational effort. For

[5] *Tree-insertion grammar (TIG)* is a constrained variant of TAG that is more amenable to computational processing (Hwa, 1998).

[6] We have been using the asynchronous case of parsing for simplicity of experimentation and presentation. However our arguments straightforwardly translate into the synchronous case of translation.

example, for nonparametric TSG induction, Cohn and Blunsom (2010) have argued for the necessity for *blocked sampling* algorithms, where a group of highly correlated variables are sampled jointly at once, therefore improving the mixing of the sampler, exploring the space of grammars more effectively, and achieving faster convergence to better solutions. As we move beyond TSG into the realm of variants of TAG, degrees of freedom of sampled derivations increase with the added adjunction operation and its richer combinatorics; therefore the need for blocked sampling strategies becomes even more dire. We provide effective samplers and efficient parsing strategies for TAG (and TIG) and compare against TSG as well as a more naive *Gibbs sampling* strategy with poorer convergence. We show improvements in parsing performance while inducing more compact syntactic representations at the same time.

Finally, in Chapter 5 we demonstrate the effectiveness of our nonparametric inference scheme by comparing nonparametric STSG against two parametric STSGs within the task of sentence compression. By achieving improvements against the parametric baselines we illustrate the merits of data-driven nonparametric inference over the space of grammars as opposed to sparse parametric inference with an a priori fixed grammar. Induced grammars on the Wikipedia data have similar performance to lexicalized parametric SCFG on the same data, however the benefit of the nonparametric approach is clear from the dramatic reduction in the number of induced parameters.

Chapter 2

Background

In order to better understand the benefits of the Bayesian nonparametric framework with respect to grammar induction, we must cover the basics of statistical machine learning (particularly generative modeling principles[1]) and grammar formalisms. In this chapter, we first motivate the Bayesian element by drawing a contrast with the key parametric estimation concepts of maximum likelihood and maximum a posteriori. We also touch upon parametric latent variable models that facilitate complex inferences that are central to grammar induction scenarios, and a portfolio of algorithms that make possible the use of these latent variable models, namely, expectation maximization, Markov chain Monte Carlo, and variational inference. Secondly, we discuss syntax and grammars. We introduce various grammar formalisms including finite-state models, context-free grammars, and tree-substitution grammars, leading up to the linguistically rich formalism of tree-adjoining grammars, all of which are crucial to follow our arguments in this thesis. Finally, we introduce the Bayesian

[1] Generative models are a type of statistical model whereby a set of distributional assumptions are made about the process by which the data is generated.

nonparametric framework and highlight the way in which it resolves the crux of the grammar induction problem, namely, model selection.

2.1 Statistical machine learning

In statistical machine learning, we are generally interested in drawing probabilistic inferences based on a dataset of observations. Roughly speaking, we make some probabilistic modeling assumptions about these observations, perform some sort of *model fitting* or *estimation* (the training phase), and use this knowledge to make predictions about unseen, novel data (the testing phase).

In the generative modeling framework, which we will use mostly throughout the thesis, our modeling assumptions are probabilistic, distributional assumptions involving the observed and unobserved variables, as well as the parameters of these distributions. Perhaps the most primitive form of this is *frequentist point-estimation*. In point-estimation, we typically have some *independent and identically distributed (iid)* observations $\mathbf{x} = \{x_1, \ldots, x_N\}$ (training examples), we make some modeling assumptions by assuming a *parametric form* for the distribution of $x \sim p(x \mid \theta)$ (we will look into *nonparametric* modeling later), and do model fitting by finding the particular model $\hat{\theta}$ that *best explains* the observations by some objective measure. Under iid conditions, the sum of our knowledge about an unseen data point x_{N+1} is the following

$$x_{N+1} \sim p(x \mid \hat{\theta}) \tag{2.1}$$

A variety of inferences can be drawn from this knowledge (including, but not limited to, the mean, mode, and variance of the distribution). Next, we will introduce two very important

point-estimation methods: maximum likelihood and maximum a posteriori estimation. Although being frequentist, and therefore fundamentally non-Bayesian, these methods help lay the groundwork for the statistical inference terminology that we will need to present the Bayesian methodology in Section 2.1.3.

2.1.1 Maximum likelihood estimation

One way to find an optimal model is to use *maximum likelihood estimation (MLE)*. MLE attempts to answer the question, what is the model $\hat{\theta}$ that best explains the data? The answer is, "the model under which the data would be most likely".

$$\hat{\theta}_{\text{MLE}} = \underset{\theta}{\operatorname{argmax}} \log p(\mathbf{x} \mid \theta) \tag{2.2}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(x_n \mid \theta) \tag{2.3}$$

Here, $p(\mathbf{x} \mid \theta)$ is known as the *likelihood*, and $\log p(\mathbf{x} \mid \theta)$ the *log-likelihood*.[2] Despite being mathematically elegant and tractable, the MLE is generally prone to *overfitting*. If we assume a highly flexible parametric family for θ, the MLE will commit too strongly to the peculiarities of the data. For instance, if we have a unigram language model, the tokens in the vocabulary that are unobserved in the data (due to, say, having a small dataset) will have zero probability under the MLE estimate.

[2]The reason log-likelihood is used is that expressions comprised of sums of logarithms are more amenable to mathematical processing than products. Since the logarithm is a monotonically increasing function the maximum is unaffected.

2.1.2 Maximum a posteriori estimation

To prevent this, we trade off the evidence from the data (the likelihood) with our expectations about what the model should look like

$$\begin{aligned} \text{posterior} &\propto \text{likelihood} \times \text{prior} \\ p(\theta \mid \mathbf{x}) &\propto p(\mathbf{x} \mid \theta) p(\theta) \end{aligned}$$

Here, $p(\theta)$ and $p(\theta \mid \mathbf{x})$ are referred to as the *prior* and *posterior* distributions of θ. The estimate that maximizes the posterior distribution as an objective is called the maximum a posteriori (MAP) estimate.

$$\begin{aligned} \hat{\theta}_{\text{MAP}} &= \operatorname*{argmax}_{\theta} \log p(\theta \mid \mathbf{x}) && (2.4) \\ &= \operatorname*{argmax}_{\theta} \{\log p(\mathbf{x} \mid \theta) + \log p(\theta)\} && (2.5) \\ &= \operatorname*{argmax}_{\theta} \left\{ \sum_{n=1}^{N} \log p(x_n \mid \theta) + \log p(\theta) \right\} && (2.6) \end{aligned}$$

Compare Equation 2.6 to Equation 2.3 and notice the extra term $\log p(\theta)$ resulting from the prior. This estimate can be more robust to sampling bias (e.g., data sparsity) as it takes into account our prior knowledge as well as the data likelihood. For example, a prior distribution for a unigram language model can be that all words are equally likely. When we combine this information with the likelihood and maximize the joint probability distribution as an objective, we obtain more robust estimates that are not overly committed to likelihood statistics alone. In the unigram language model example, this robustness manifests as *smoothing* ragged MLE estimates (turning zero probabilities into small yet positive probabilities).

2.1.3 Bayesian modeling

Since the posterior distribution summarizes our overall knowledge about the θ parameter — What are its most probable values (*modes* of the distribution)? What value does it take on average (*mean* of the distribution)? How much uncertainty is there about its value (*variance* of the distribution)? — there is no good reason to commit to the single maximum of this distribution (especially, say, if the distribution is very multi-modal), given that there are mathematical ways to use the totality of this information.

The Bayesian modeling framework allows us to use the entire posterior distribution by treating the parameters θ not as fixed unknown values (as in MLE and MAP) but as random variables. We assume a prior distribution for θ, turning it into what we will call a *latent variable*, and use its posterior distribution in drawing any inferences. For instance, again, if we are interested in making a prediction about a novel x_{N+1} we can take into account the whole spectrum of θs by integrating over the posterior distribution:

$$x_{N+1} \sim \int p(x|\theta)p(\theta|\mathbf{x})d\theta$$

Compare this to Equation 2.1.

We might still wish to have certain parameters be treated as fixed unknown values,[3] especially in defining parameters for prior distributions. There are quite a few ways to set up notation for such models. We will follow Bishop's (2007) graphical modeling notation and make use of three categories of variables:

1. Observed variables $\mathbf{x} = \{x_1, \ldots, x_N\}$

[3] In this thesis, we will use the term Bayesian to refer to *empirical Bayesian* models, that is, Bayesian models with additional unknown parameters to be point-estimated, as opposed to *fully Bayesian* models in which all parameters are treated as random variables (Blei et al., 2003).

2. Latent (hidden, unobserved) variables $\mathbf{z} = \{z_1, \ldots, z_M\}$

3. Parameters $\theta = \{\theta_1, \ldots, \theta_K\}$

Observed and latent variables are variables that we model as stochastic. Observed variables correspond to data; latent variables can be thought of as auxiliary variables that we use to describe the *generative process* (by which the observed variables are generated) and its interior mechanisms. They generally serve to represent associations (such as clusterings, assignments, features) that are pertinent to the prediction task at hand. Parameters are fixed (non-random) variables that typically define distributions of various other random variables. They can be known or unknown; when they are unknown, point-estimation based inference (MLE or MAP) is used to predict their values. Sometimes it is useful to turn a parameter into a latent variable by assuming a prior distribution for it and use posterior inference to estimate its values (thereby making the model deeper, more hierarchical). The parameters that control the prior distributions of these variables are called *hyper-parameters*.

Gaussian mixture model example (GMM)

A simple example of this type of Bayesian model is a Gaussian mixture model (GMM) which is typically used to cluster real-valued observations. We assume K clusters, each of which is a Gaussian model (with the well-known density function denoted as $\mathrm{N}(\cdot \mid \mu_k, \sigma^2)$ where μ_k is the mean and σ^2 is the variance of the distribution), observations $x_1, \ldots, x_N$, latent variables $z_1, \ldots, z_N$ where z_n determines the assignment of x_n to a cluster. It is mathematically convenient to treat z_n as being a K-dimensional *indicator vector* such that $z_{nk} = 1$ if and only if x_n is assigned to the kth cluster and 0 otherwise. The parameters of this model

are σ^2 and $\mu_1, \ldots, \mu_K$ that determine the spread and center of each Gaussian model; moreover, we have parameters that define the prior probability of using each cluster $\pi_1, \ldots, \pi_K$ (a multinomial model where $\sum_{k=1}^{K} \pi_k = 1$). Therefore the generative model is the following:

For each $n = 1, \ldots, N$

- pick a cluster $z_n \sim \text{multinomial}(\pi_1, \ldots, \pi_K)$,
- generate an observation $x_n \sim N(\mu_k, \sigma^2)$ if $z_{nk} = 1$ and $z_{nj} = 0$ for all $j \neq k$.

We write the multinomial probability distribution as $p(z_n \mid \pi) = \prod_{k=1}^{K} \pi_k^{z_{nk}}$.

2.1.4 Expectation maximization

Our previous recipes for inference, MLE and MAP, do not straightforwardly apply to models with latent variables. Under latent variable conditions, often there is no closed-form, analytical expression for $p(\mathbf{x} \mid \theta) = \int p(\mathbf{x}, \mathbf{z} \mid \theta) d\mathbf{z}$. For example, for the GMM, we have

$$\log p(\mathbf{x} \mid \pi, \mu, \sigma^2) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k N(x_n \mid \mu_k, \sigma^2)$$

an expression difficult to solve for the optimal parameters π, μ, σ^2. The difficulty stems from the *log-sum expression*; if the latent $\mathbf{z}$ were observed, we would be able to apply simple MLE principles. This is the canonical *missing data* situation (interpreting the latent variables as part of the data variables) and *expectation maximization (EM)*, the most ubiquitous method of inference under latent variables, straightforwardly applies (Dempster et al., 1977).

This algorithm, allows us to optimize for $\hat{\theta}_{\text{MLE}}$ without committing to particular values

of the latent variables, but by summing over their posterior distributions. The general principle is to iteratively improve $\hat{\theta}$ by solving the following equation.

$$\hat{\theta}^{(t+1)} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{p(\mathbf{z}|\mathbf{x},\hat{\theta}^{(t)})}[\log p(\mathbf{x},\mathbf{z} \mid \theta)] \tag{2.7}$$

At iteration $t+1$ we have the estimate from previous iteration $\hat{\theta}^{(t)}$ which we update by first computing the expectation of the *sufficient statistics* involving the latent variables (expectation / E-step) and then re-estimating $\hat{\theta}$ (maximization / M-step). This iterative procedure has been shown to converge to the local $\hat{\theta}_{\text{MLE}}$ solution.

To illustrate, for the GMM, we have $\theta = \{\mu, \pi\}$ (assume for simplicity that σ^2 is known). We write

$$\begin{aligned}
\log p(\mathbf{x},\mathbf{z} \mid \theta) &= \log \prod_{n=1}^{N}\prod_{k=1}^{K} \left[\pi_k N(x_n \mid \mu_k, \sigma^2)\right]^{z_{nk}} \\
&= \sum_{k=1}^{K}\left(\sum_{n=1}^{N} z_{nk}\right)\log \pi_k + \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \log N(x_n \mid \mu_k, \sigma^2)
\end{aligned}$$

Since this is a linear expression of z_{nk} we can simply take its expectation with respect to $p(\mathbf{z} \mid \mathbf{x}, \hat{\mu}^{(t)}, \hat{\pi}^{(t)})$ and optimize with respect to the parameters giving us the next parameter estimates for iteration $t+1$

$$\hat{\pi}_k^{(t+1)} = \frac{\sum_{n=1}^{N} \mathbb{E}[z_{nk}]}{N}, \qquad \hat{\mu}_k^{(t+1)} = \frac{1}{\sum_{n=1}^{N} \mathbb{E}[z_{nk}]} \sum_{n=1}^{N} \mathbb{E}[z_{nk}]\, x_n \tag{2.8}$$

where the expected sufficient statistics involving the latent variables are

$$\mathbb{E}[z_{nk}] = \frac{\hat{\pi}_k^{(t)} N(x_n \mid \hat{\mu}_k^{(t)}, \sigma^2)}{\sum_{j=1}^{K} \hat{\pi}_j^{(t)} N(x_n \mid \hat{\mu}_j^{(t)}, \sigma^2)} \tag{2.9}$$

See Bishop (2007) for the derivation.

2.1.5 Markov chain Monte Carlo and Gibbs sampling

We have seen EM as a method of inference for models with latent variables. A very viable alternative that applies to even more general latent variable settings is *Markov chain Monte Carlo (MCMC)*. In this method, instead of using the expected values of latent variables or sufficient statistics thereof (as we do in EM), we represent the posterior information about the latent variables by maintaining samples from their corresponding posterior distributions. In general, it can be intractable to draw samples from any posterior distribution, however MCMC provides a way of designing a *Markov chain* for any distribution such that, in the limit, samples from this Markov chain converge to the desired distribution. *Metropolis-Hastings (MH)* algorithm is the most widely applicable recipe for simulating such a Markov chain (Hastings, 1970; Metropolis et al., 1953). For any $p(z)$ which is assumed to be evaluated easily up to the normalization constant, we simulate a Markov chain $z^{(t)}$ for $t = 1, 2, \ldots$ by, at each step, drawing a sample z^* from a *proposal distribution* $q(z \mid z^{(t)})$, which we choose so that it is easy to draw samples from, and accept the new sample z^* as $z^{(t+1)}$ if

$$u < \frac{q(z^{(t)} \mid z^*)p(z^*)}{q(z^* \mid z^{(t)})p(z^{(t)})}$$

where $u \sim$ Uniform(0,1). If z^* is rejected, set $z^{(t+1)} = z^{(t)}$.

A simple special case of the MH algorithm which also has broad applicability but may have subpar convergence is *Gibbs sampling* (Geman and Geman, 1984). In this method, we cycle though the latent variables and sample from their conditional distributions given the rest of the sampled values, always accepting the new samples. Gibbs sampling is a special case of the MH algorithm because if we think of the conditional distribution as being the proposal distribution q, we can show that the probability of acceptance is 1 (Bishop, 2007).

This algorithm is especially convenient when distributions are from the *exponential family* and *conjugacy relationships* hold between priors and likelihoods, so that the conditional distributions are familiar and easy to sample from.

To illustrate, for the GMM model, we have conditional posteriors defined as

$$p(z_n \mid x_n, \mu, \pi, \sigma^2) \propto p(x_n \mid z_n, \mu, \sigma^2) p(z_n \mid \pi)$$

or, equivalently,

$$p(z_{nk} = 1 \mid x_n, \mu, \pi, \sigma^2) \propto \pi_k N(x_n \mid \mu_k, \sigma^2)$$

a multinomial model, which is convenient to sample from. We represent the posterior distribution of latent $\mathbf{z} = \{z_n\}$ by drawing samples from this distribution. Note the resemblance to Equation 2.9.

To demonstrate the idea of inferring parameters by turning them into latent variables, let us deepen our hierarchy by adding that $\pi \sim \text{Dirichlet}(\alpha)$ where $\alpha = \{\alpha_1, \ldots, \alpha_K\}$ is a fixed constant vector. Now in addition to sampling $\mathbf{z}$, we can also sample π from its conditional distribution given the rest of the variables:

$$\begin{aligned} p(\pi \mid \mathbf{x}, \mathbf{z}, \mu, \sigma^2, \alpha) &\propto p(\mathbf{z} \mid \pi) p(\pi \mid \alpha) \\ &= \left[\prod_{n=1}^{N} p(z_n \mid \pi) \right] p(\pi \mid \alpha) \\ &\propto \left[\prod_{k=1}^{K} \pi_k^{(\sum_{n=1}^{N} z_{nk})} \right] \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} \\ &= \prod_{k=1}^{K} \pi_k^{(\sum_{n=1}^{N} z_{nk} + \alpha_k - 1)} \end{aligned}$$

meaning that, conditionally, $\pi \sim \text{Dirichlet}(\{\sum_{n=1}^{N} z_{nk} + \alpha_k\})$, the conjugate posterior which, given our samples of $\mathbf{z}$ being readily instantiated, is easy to sample from.

2.1.6 Variational inference

Theoretically speaking, MCMC will converge on the true posterior as long as it is executed for a long enough time. In practice, poor MCMC design (bad choice of proposal distribution, sampling highly correlated variables independently, etc.) can cause this time requirement to become impractical. On the contrary, in NLP, we often need inference techniques that are computationally lean and highly scalable since our models may have to be re-trained many times over for specific domains, applications, datasets; moreover, our predictions must be computed in real-time for real applications such as syntactic parsing, speech recognition, and machine translation. For this reason one might prefer an algorithm such as EM, which uses expected sufficient statistics, to MCMC, which uses many samples to represent the same kind of information. However we know that EM applies to a limited class of models where the expectation of the log-joint-likelihood can be computed easily with respect to the posterior distribution of the latent variables (see Equation 2.7). Unfortunately, this is rarely the case in deep and hierarchical Bayesian models (Blei et al., 2003). The method of *variational inference (VI)* was proposed as a generalized alternative that relaxes this very requirement of EM (Jordan et al., 1999).

In many Bayesian models, the posterior distribution of the latent variables can exhibit undesirable interactions between some variables that make the EM computation analytically intractable. In VI, instead of using this complex posterior, we assume a *simpler* family of alternative posterior distributions $q(\mathbf{z})$ where the undesirable interactions are strictly prohibited[4], and search for the member of this family that best approximates the true pos-

[4]This decoupling of variables is typically achieved by assuming *factorized* distributions.

terior,

$$\hat{q}(\mathbf{z}) \approx p(\mathbf{z} \mid \mathbf{x}, \theta^{(t)})$$

by minimizing the *KL-divergence* between the two. We then take the necessary expectation of the log-joint-likelihood with respect to $\hat{q}(\mathbf{z})$ (Equation 2.7) and update the parameters as in the M-step of EM. Finding the best $\hat{q}(\mathbf{z})$ and computing expectations under it can be thought of as our new E-step, and is typically achieved by assuming a parametric family $q(\mathbf{z} \mid \gamma)$ with *variational parameters* γ and solving for the $\hat{\gamma}$ that maximizes the so-called *variational lower bound* $\mathbb{L}[\mathbf{x}, \theta, \gamma]$

$$\log p(\mathbf{x} \mid \theta) \quad = \quad \mathbb{L}[\mathbf{x}, \theta, \gamma] + \mathrm{KL}[\, q(\mathbf{z} \mid \gamma) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta) \,]$$

which at the same time leads to minimizing the aforementioned KL-divergence.[5] Now we alternate between optimizing the lower bound with respect to γ (E-step) and θ (M-step).

The lower bound further factorizes into the familiar EM quantity (2.7) and the entropy of the variational distribution as follows

$$\mathbb{L}[\mathbf{x}, \theta, \gamma] = \mathbb{E}_{q(\mathbf{z}|\gamma)}[\log p(\mathbf{x}, \mathbf{z} \mid \theta)] + \mathbb{H}[q(\mathbf{z} \mid \gamma)]$$

See the appendix of Blei et al. (2003) for the derivation of these equalities and how they relate to a famous inequality called *Jensen's inequality*.

Let us look at the GMM where $\pi \sim \mathrm{Dirichlet}(\alpha)$, and infer π and $\mathbf{z}$ by using the VI

[5]Note that in the E-step of EM, we set $\hat{q}(\mathbf{z}) = p(\mathbf{z} \mid \mathbf{x}, \theta)$ and make the KL-divergence vanish. The variational lower bound is maximized at the data log-likelihood.

technique. Assuming *factorized* variational families

$$\begin{aligned} q(\pi, \mathbf{z}) &= q(\pi) \prod_{n=1}^{N} q(z_n) \\ &= \text{Dirichlet}(\pi \mid \gamma) \prod_{n=1}^{N} \text{multinomial}(z_n \mid \phi_n) \end{aligned}$$

with variational parameters $\gamma = \{\gamma_k\}, \phi = \{\phi_{nk}\}$. Writing out the variational lower bound and maximizing it analytically with respect to γ, ϕ we arrive at the following update equations

$$\gamma_k = \alpha_k + \sum_{n=1}^{N} \phi_{nk}, \qquad \phi_{nk} \propto \frac{\exp \psi(\gamma_k)}{\exp \psi(\sum_k \gamma_k)} \mathrm{N}(x_n \mid \mu_k, \sigma^2)$$

where ψ (*Digamma function*) is the derivative of the log-Γ function. In every E-step, we alternate between these updates until convergence. See Appendix A for the derivation.

Note the similarity to EM update equations (Equations 2.8, 2.9). VI for Dirichlet-multinomial models requires a mere update to EM where we simply replace the expected counts $\mathbb{E}[z_{nk}]$ with

$$\frac{\exp \psi(\alpha_k + \sum_n \mathbb{E}[z_{nk}])}{\exp \psi(\sum_k (\alpha_k + \sum_n \mathbb{E}[z_{nk}]))} \mathrm{N}(x_n \mid \mu_k, \sigma^2)$$

a functional form that has been shown to fight overfitting by regularizing the expected counts obtained from sparse data (Johnson, 2007).

2.2 Grammar formalisms

A stochastic grammar allows (conditional) probabilities to be associated with the independent decisions, the probability of the overall derivation being the product of these. For example, context-free grammar (CFG) allows combination of strings according to a

finite taxonomy of nonterminals; probabilistic CFG additionally provides a probability distribution over the concatenation possibilities, which induces a probability distribution over derived trees and strings.

The sequence of probability values, taken together, indexes a parametric family of grammars. Given the observations and a grammar formalism, the inference task is to reason about these unknown parameters: about the number and nature of them as well as their particular values. This is typically achieved by searching for the *best* setting of the parameters (as evidenced by the observations) or by computing posterior distributions for them.

Grammar formalisms for NLP tasks have steadily progressed in intricacy as researchers have searched for a better match between grammar formalism and the empirical language data being modeled while attempting to maintain algorithmic feasibility. In caricature, the trend has been from finite-state models to phrase-structure-models, to tree-based models.

2.2.1 Finite-state models and context-free grammars

Finite-state models (FSM) characterize a language based on a finite typology of the preceding context of each word. In n-gram models, for instance, the types are given by the $n-1$ preceding words. For each possible context type there is a probability distribution over following words, which constitute the parameters of the model. The modeling power of the formalism is limited by the fact that it eschews the single most robust fact about natural language known for millennia, that sentences have hierarchical structure. The hierarchical nature of language means that dependencies among words can be at an arbitrary distance; phrases of differing length intervening between two words do not, to first-order, change the dependency relationship. For example, we would like to statistically model the relation

between the words "boys" and "were" in the sentence "the boys were expelled" (cf. "the boys was expelled") similarly to "the boys who defaced the statue were expelled" (cf. "the boys who defaced the statue was expelled"). The independence assumptions of *n*-gram models, that words are independent of words more than a fixed distance prior, make this difficult to do.

Phrase-structure grammars such as context-free grammars (CFG) are better able to model these kinds of relations, since the intervening material can be characterized not merely as a sequence (of unspecified length) of characters, but as a single phrase in a hierarchical structure. Probabilistic context-free grammars apply the typology not to prior contexts of a word but to contiguous subsequences of phrases. Alternatively, we can think of small trees of a parent type and its immediate children such as the familiar S $\rightarrow$ NP VP, etc. Each type can be given a probability distribution over all of the possible right-hand-side sequences; these probabilities, again, are the parameters of the model.

2.2.2 Tree-substitution grammars

The problems with probabilistic CFGs are well known. For instance, they eliminate much of the lexical grounding that makes *n*-gram models so effective. The independence assumptions made by CFG have thus proven too permissive, such that grammar refinement techniques — *lexicalization* (Magerman, 1995; Charniak, 1997; Collins, 2003), symbol-splitting (Klein and Manning, 2003a; Petrov et al., 2006; Liang et al.), etc. — have been developed with the specific purpose of repairing these problems.

An alternative way to model context-free string languages is to use tree-substitution grammars (TSG), which replace the single-level parent-and-child elementary trees of CFG

with a set of multi-level trees, thereby relaxing the independence assumptions of CFG.

A TSG is a 4-tuple, $G = (T, N, S, E)$, where T is a set of *terminal symbols* (lexical items), N is a set of *nonterminal symbols*, $S \in N$ is the distinguished *root nonterminal*, and E is a set of *elementary trees*. The elementary trees are tree fragments of depth ≥ 1,[6] where each internal node is labeled with a nonterminal and each leaf is labeled with either a terminal or a nonterminal. Nonterminal leaves are called *frontier nodes* and form the substitution sites in the generative process of creating trees with the grammar.

A TSG *derivation* generates a tree by starting with the root symbol and substituting it with an elementary tree that has the same symbol at the root (this operation is called *substitution*), and then continuing to substitute frontier nonterminals with elementary trees until there are no remaining frontier nonterminals. Unlike CFGs, a syntax tree may not uniquely specify the derivation, as illustrated in Figure 2.1; we refer to this ambiguity as *derivational ambiguity*.

A *probabilistic* TSG, like a probabilistic CFG, assigns a probability to each production (elementary tree) such that every nonterminal c is associated with a multinomial distribution over elementary trees e that may substitute for that symbol at a frontier node. We have,

$$p(e \mid \text{root}(e)) > 0, \qquad \sum_{e \text{ s.t. root}(e) = c} p(e \mid c) = 1$$

where $\text{root}(\cdot)$ is the function that returns the nonterminal that labels the root of an elementary tree. The probability of a derivation $\mathbf{e}$ comprised of D elementary trees $e_1, \ldots, e_D$ is

[6] Elementary trees of depth 1 correspond to productions in a CFG.

the product of the probabilities of its elementary trees,

$$p(\mathbf{e}) = \prod_{d=1}^{D} p(e_d \mid \text{root}(e_d))$$

and the probability of a tree t is the sum of the probabilities of its derivations

$$p(t) = \sum_{\mathbf{e} \text{ derives } t} p(\mathbf{e})$$

By structuring the grammars in the model class differently, and hopefully in ways more suited to the particularities of the language phenomena being modeled, a more accurate model can be generated with fewer parameters (Goodman, 1998; Bod, 1998). The greater expressivity of TSG allows for grammars that include much larger elementary units, for fully structured phrases complete with lexical material — idioms such as "cut the mustard" or light verb constructions such as "take advantage of" or grammatically determined collocations as in the perfect progressive construction "has been VB". Since these word sequences are overrepresented as compared to their parts, they can be assigned higher probability directly by a formalism structured as a TSG (Figure 2.1).

2.2.3 Tree-adjoining grammars

Statistical grammar induction algorithms use repetition of particular structures in training data as an indication of their probability value in the grammar. Thus to the extent that the formalism allows for breaking up of the hierarchical structure of a sentence into parts that are more often repeated (relative to their expected base rate of occurrence), the formalism allows for better modeling of the training data. This argues for adding an additional combinatory operation in the grammar formalism, *adjunction* (Figure 2.2). The adjunction operation, which has been actively investigated in computational linguistics for

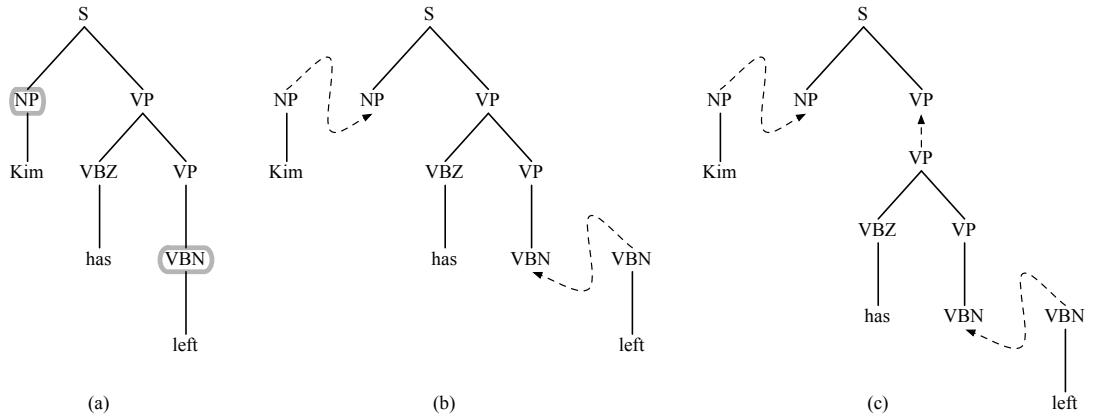


Figure 2.1: (a) The syntax tree for "Kim has left". Substitution points are marked with gray circles. (b) The corresponding TSG elementary trees used to derive this tree (A TSG derivation). Dashed arrows mark the substitutions performed. (c) An alternative derivation for the same syntax tree. Notice that exponentially many potential derivations underlie any syntax tree.

several decades now starting with the pioneering work of Joshi et al. (1975), allows for the splicing in of material in the middle of a TSG-like elementary tree. This allows, as we will argue below, the capturing of more re-use of elementary trees in training data, and the potential for better modeling of the data.

Much like TSG, the formalism of tree-adjoining grammars (TAG) is also a tree-generating system. A TAG is a 5-tuple, $G = (T, N, S, I, A)$, where T is a set of terminal symbols, N is a set of nonterminal symbols, $S \in N$ is the distinguished root nonterminal, and I is a set of *initial trees* that are TSG-style elementary trees, and A is a set of *auxiliary trees* – tree fragments that are similar to initial trees, except substitutions occur at all but one frontier node, called the *foot node*; by convention, the foot node is annotated with an asterisk (*); the label of the foot node must be identical to the label of the root node. The trees in $I \cup A$ are the elementary trees of TAG (some examples of initial and auxiliary trees are given in Figure 2.2). We call an elementary tree a c-type elementary tree if its root is labeled by the

nonterminal c.

In addition to substitution, a TAG derivation involves a secondary composition operation called adjunction by which an auxiliary tree is *spliced in* to any node in a partial derivation[7] (notice the handling of auxiliary tree β in Figure 2.2). The derivation is complete once a substitution or adjunction decision is made about every node that is a potential site for either operation. All internal nodes except those marked specifically for *null adjunction* are valid adjunction sites.

In order to define probabilistic TAG, we must define probabilities for the derivational events involving substitution and adjunction. In general, substitutions are handled the same way as probabilistic TSG where each elementary tree has its own probability. Traditionally, adjunction probabilities are defined such that every *node* (as opposed to every nonterminal) in every initial or auxiliary tree is associated with a multinomial distribution over auxiliary trees that may be adjoined at that node (and the *none* option, that is, no adjunction at that node) (Resnik, 1992; Schabes, 1992). For every node η labeled by the nonterminal c and every auxiliary tree β that is c-type, we have

$$p(\beta \mid \eta) \geq 0, \quad p(\text{none} \mid \eta) \geq 0,$$
$$p(\text{none} \mid \eta) + \sum_{\beta \text{ s.t. root}(\beta) = c} p(\beta \mid \eta) = 1$$

Alternatively, one can have a dedicated Bernoulli random variable for the decision to adjoin or not (with probability of adjunction being $\mu_\eta \in [0,1]$), and a multinomial distribution over the auxiliary trees given that an adjunction occurs

$$p(\beta \mid \eta) \geq 0, \quad \sum_{\beta \text{ s.t. root}(\beta) = c} p(\beta \mid \eta) = 1 \tag{2.10}$$

[7] A partial derivation is a derivation that is incomplete.

such that adjunction is defined as a double-event whose probability is

$$\mu_\eta p(\beta \mid \eta) \tag{2.11}$$

Due to its additional functionality, TAG generates *mildly context-sensitive* string languages (Joshi et al., 1975). This increased generative power of adjunction allows important linguistic dependencies to be expressed *locally* in the grammar, that is, within the relevant elementary trees, rather than by many specialized complex rules, as exemplified in Figure 2.2.

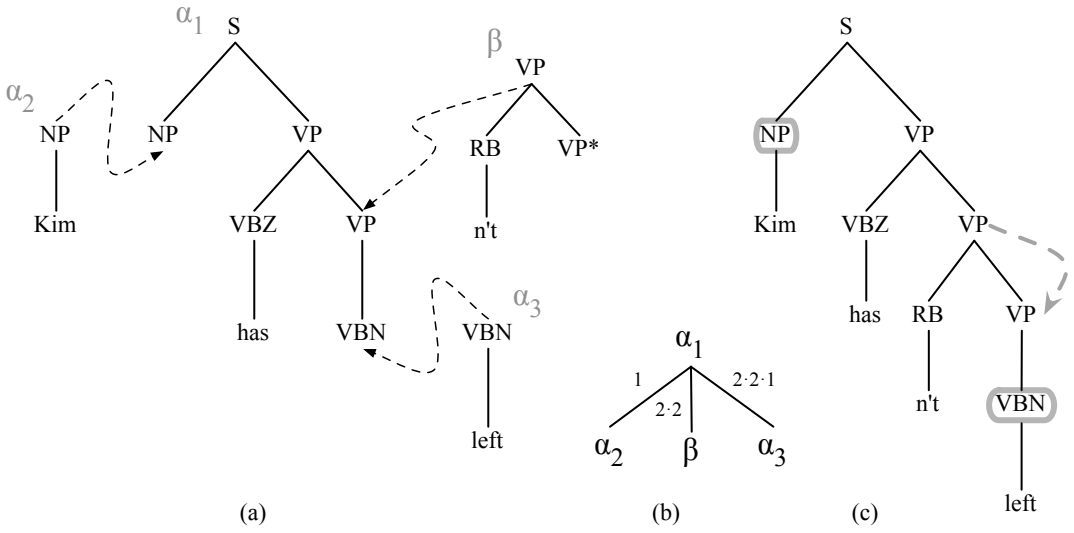


Figure 2.2: Various representations of TAG derivations. (a) Explicit representation of the elementary trees (marked with Greek letters for reference) showing the derivation steps. (b) Traditional *derivation tree* capturing the derivation. (c) Our flat representation of the derivation tree, depicted as the derived tree (that is, the syntax tree itself) with simple annotations: Circles indicate substitution points, dashed arrows indicate adjunction.

The major problem with TAG so far has been its even more daunting model selection problem, as well as its computational complexity. When using TAG, one must reason about not only the substitution operation but also auxiliary trees and their adjunctions, causing a

combinatorial explosion of hypotheses about deriving the data. Furthermore, parsing with TAG has complexity $O(n^6)$ for a sentence of length n, as opposed to $O(n^3)$ for context-free formalisms such as CFG and TSG (Schabes, 1992). This is prohibitive not only for testing but also for many training algorithms (such as EM) that rely on iterative parsing of the data. Tree-insertion grammars (TIG) are a constrained form of TAG, where the foot node of an auxiliary tree necessarily falls to the left (or right) of all adjoined material (Schabes and Waters, 1995; Hwa, 1998). (Auxiliary tree β in Figure 2.2, as well as most auxiliary trees of English, conform to this constraint.) Although TIG supports adjunctions, it remain context-free and therefore parsing with TIG is highly scalable.

2.3 Statistical grammar induction

In this thesis, we will focus on the supervised training setup for grammar induction where we have a training corpus of parsed sentences (or pairs of sentences) which we use to predict syntactic analyses for unseen sentences. If we assume a parametric family (e.g., the set of rules that define a CFG) our inference problem becomes one of *model estimation*: only the values of the parameters are to be determined. If we start with no assumptions, the inference problem is the dual problem of *model selection and estimation*: we must determine the parameters and their values simultaneously. Recently, nonparametric Bayesian approaches have been useful to remedy this more difficult problem in a non-heuristic, principled way. Let us first review the more traditional parametric approaches.

2.3.1 Parametric models

For CFGs, this is a straightforward task, which can be solved using MLE or MAP principles: Assuming the grammar is defined by multinomial parameters $\{\theta_{kr}\}$ where k indexes the set of nonterminals and r indexes the set of potential CFG rules expanding this nonterminal, we have the MLE estimate

$$\hat{\theta}_{kr} = \frac{\text{Count}(r)}{\text{Count}(k)}$$

where $\text{Count}(\cdot)$ is a function that counts the number of occurrences in the training data.

When we move to the TSG (or TAG) domain, we are faced with *derivational ambiguity*: there can be many derivations underlying a training tree. Therefore our inference problem is now a latent variable problem and our generative process is as follows.

For each training tree t_n, $n = 1, \ldots, N$

- sample a derivation $e_n \sim \text{TSG}(\theta)$,
- given the derivation, the tree t_n is determined.

The $\text{TSG}(\cdot)$ distribution is defined as follows:

$$p(e_n) = \prod_{k=1}^{K} \prod_{r=1}^{N_k} \theta_{kr}^{z_{nkr}}$$

where N_k is the number of TSG rules (elementary trees) for nonterminal k and z_{nkr} is the number of times rule r of nonterminal k was used in derivation e_n. Similar to the GMM example, we can use EM to estimate θ:

$$\hat{\theta}_{kr} = \frac{\sum_{n=1}^{N} \mathbb{E}[z_{nkr}]}{\sum_{r=1}^{N_k} \sum_{n=1}^{N} \mathbb{E}[z_{nkr}]}$$

The Inside-Outside algorithm (Lari and Young, 1990; Eisner, 2003) can be used to compute the necessary expected sufficient statistics $\mathbb{E}[z_{nkr}]$.

Unfortunately EM has the shortcoming of strongly overfitting the data by arbitrarily preferring larger elementary trees (Prescher et al., 2003). To prevent this behavior, we can assume an appropriate sparse prior so that $\theta_k \sim \text{Dirichlet}(\alpha_k)$ where $\alpha_k = \{\alpha_{k1}, \ldots, \alpha_{kN_k}\}$, and use VI to integrate over the parameters, in which case we have the E-step update equations

$$\hat{\theta}_{kr} = \frac{\exp \psi \left(\alpha_{kr} + \sum_{n=1}^{N} \mathbb{E}[z_{nkr}] \right)}{\exp \psi \left(\sum_{r=1}^{N_k} (\alpha_{kr} + \sum_{n=1}^{N} \mathbb{E}[z_{nkr}]) \right)}$$

We will use this VI training algorithm in our synchronous TSG experiments as an alternative to EM and MCMC (See Section 5.1.2).[8]

2.3.2 Nonparametric models

The above solutions to the problem of model estimation are well-studied, yet the real inference problem that we will address in this thesis is *model selection*: how to determine the grammar (the set of elementary trees themselves, not their probability values) in the first place. This is a challenge because brute-force enumeration will lead to prohibitive training times, and heuristic grammar extraction techniques use specific biases that hurt predictive performance. We will harness the nonparametric Bayesian model selection techniques to search over the space of all grammars and find a representation that is best supported by the data.

[8] Alternatively, for the same model we can use Gibbs sampling (sample $\{e_n\}$ given $\{\theta_k\}$, sample $\{\theta_k\}$ given $\{e_n\}$) or Metropolis-Hastings if we use a *collapsed* model where $\{\theta_k\}$ are marginalized out (Johnson and Griffiths, 2007).

One important mathematical construct we will exploit is the *Dirichlet process* (DP). DP is a stochastic process commonly used in Bayesian nonparametrics for clustering data where the number of clusters is unknown (so far in our presentation, for example in the GMM example, the number of clusters K was known and fixed). This situation arises frequently in NLP outside of grammar induction. For example, in *topic analysis* we may not know ahead of time how many topics exist in a corpus of documents without reading the whole set of documents. Similarly, for applications such as *named entity recognition* or *coreference resolution* we may not know how many types of entities are used or how many unique entities are being referred to. DP, by using an adaptive *rich-gets-richer* process whereby large clusters get ever larger and a large number of small clusters remain small, can infer the optimal number of clusters for a particular dataset.

Dirichlet process Gaussian mixture model example

To illustrate, we can think of a Dirichlet process Gaussian mixture model (DP-GMM) in which the number of clusters K is not fixed and will be inferred from the data. This is a well-studied model (Neal, 2000) which is sometimes referred to as an *infinite* GMM since it hypothesizes an arbitrary number of clusters; the parametric complexity of the model is adaptive (to the dataset). The DP-GMM has the following generative process:

Sample $G \sim \mathrm{DP}(\alpha, P_0)$. For each $n = 1, \ldots, N$

- sample a cluster mean $\mu_n \sim G$,
- sample $x_n \sim \mathrm{N}(x_n \mid \mu_n, \sigma^2)$.

The *concentration parameter* $\alpha > 0$ controls the expected number of clusters (large α lead to large number of clusters); *the base distribution* P_0 is a distribution that generates cluster means μ, say $P_0 = \mathrm{N}(\mu \mid \mu_0, \sigma_0^2)$. Using the DP as a prior puts the μ_n in the following conditional relationship to the previous samples

$$\mu_n \mid \mu_1, \ldots, \mu_{n-1} \sim \frac{\mathrm{Count}_{<n}(\phi_c)\,\delta(\phi_c) + \alpha P_0}{n-1+\alpha} \tag{2.12}$$

$$= \left(\frac{\mathrm{Count}_{<n}(\phi_c)}{n-1+\alpha}\right)\delta(\phi_c) + \left(\frac{\alpha}{n-1+\alpha}\right)P_0 \tag{2.13}$$

where $\{\phi_c\}$ represent the unique cluster means among $\mu_{<n} = \{\mu_1, \ldots, \mu_{n-1}\}$, $\mathrm{Count}_{<n}(\phi_c)$ is an operator that counts how many times ϕ_c was sampled in $\mu_{<n}$, and $\delta(\cdot)$ is the *point-mass distribution* that puts all of its probability mass on a single point. We either sample one of the ϕ_c that were previously sampled (with probability proportional to the number of times they were sampled) or instantiate a new cluster by sampling a new ϕ from P_0 (with probability proportional to α). Here, the rich-gets-richer dynamics are seen mathematically: the more we use ϕ_c the more likely we become to use it again in future samples. Moreover, the larger the concentration parameter α the more the process is willing to create a large number of small clusters. Note that this is an *exchangeable* process, in that the order of the μ does not matter, since the distribution depends only on the overall counts $\mathrm{Count}_{<n}(\phi_c)$.

The basic Gibbs sampler for the DP-GMM uses this above property, steps through the data points x_n and updates their cluster assignment variables z_n by either assigning them to an already existing cluster or letting x_n start its own fresh cluster (Neal, 2000). We update

each z_n as in the following:

$$\begin{aligned} p(z_n = c \mid \mathbf{z}_{-n}, \mathbf{x}, \phi) &\propto \frac{\text{Count}_{-n}(\phi_c)}{N-1+\alpha} \text{N}(x_n \mid \phi_c, \sigma^2) \\ p(z_n = \text{new} \mid \mathbf{z}_{-n}, \mathbf{x}, \phi) &\propto \frac{\alpha}{N-1+\alpha} \int \text{N}(x_n \mid \phi, \sigma^2) P_0(\phi) d\phi \end{aligned}$$

where $\mathbf{z}_{-n}$ is the set $\mathbf{z} - z_n$ and $\text{Count}_{-n}(\phi_c)$ counts how many x_n are assigned to ϕ_c in $\mathbf{z}_{-n}$.

Nonparametric grammar induction

These ideas translate elegantly to grammar induction settings. We can apply the DP rich-gets-richer dynamics to TSG derivations: the more we use an elementary tree, the more we will be likely to use it again (Cohn et al., 2009; Post and Gildea, 2009). Posterior inference breaks up training trees into an arbitrary set of elementary trees with robust frequencies, effectively searching over the space of all grammars to find representations that are best supported by the data. The generative model will be as follows:

For each $c = 1, \ldots, C$ nonterminal label

- sample $G_c \sim \text{DP}(\alpha_c, P_0(\cdot \mid c))$.

For each training tree t_n, $n = 1, \ldots, N$

- sample a derivation $e_n = \{e_{n1}, \ldots, e_{nD}\} \sim \{G_c\}$ such that elementary tree e_{nd} with root nonterminal label c is sampled $e_{nd} \sim G_c$,
- given the derivation, the tree t_n is determined.

In this model we have C independent DPs for every nonterminal label generating the data. $P_0(\cdot \mid c)$ is itself a generative process for sampling elementary trees with root label c. Now,

a straightforward Gibbs sampling algorithm can be used to explore posterior derivations underlying the training trees (Cohn et al., 2009; Post and Gildea, 2009), and converge on a compact set of elementary trees with robust re-use statistics throughout these derivations. We step through the nodes of the training trees and make a sampling update to the current derivation introducing or removing a substitution point (introducing the subtitution point breaks an otherwise "merged" elementary tree e_M into two elementary trees e_A, e_B). By the exchangeability property we have

$$p(\text{merge}) = \frac{\text{Count}(e_M) + \alpha_{c_M} P_0(e_M \mid c_M)}{\text{Count}(c_M) + \alpha_{c_M}} \tag{2.14}$$

$$p(\text{split}) = \frac{\text{Count}(e_A) + \alpha_{c_A} P_0(e_A \mid c_A)}{\text{Count}(c_A) + \alpha_{c_A}} \times \frac{\text{Count}(e_B) + \mathbb{1}[e_A = e_B] + \alpha_{c_B} P_0(e_B \mid c_B)}{\text{Count}(c_B) + \mathbb{1}[c_A = c_B] + \alpha_{c_B}} \tag{2.15}$$

$\text{Count}(e')$ is the number of the occurrences of e' throughout the rest of the training derivations except the part that is "covered by" e_M, and $\mathbb{1}[\cdot]$ is the indicator function that takes values 0 or 1 depending on the truth value of its argument. In future chapters we will present a Gibbs sampler for model selection of TAG that operates based on the same principles (See section 4.2).

The Gibbs sampler makes changes to the current derivation through making changes locally at a node. In local methods however, one often has to step through a sequence of low probability states to reach a high probability state, which hurts the exploration of the sampler. Global methods such as *blocked sampling* by which a group of variables that are highly correlated with each other are sampled jointly can remedy this problem. Cohn and Blunsom (2010) applied this idea to TSG induction. They sample whole derivations for one training tree all at once given the rest of the corpus. The *infinite TSG* implied by the

rest of the corpus is the natural distribution to sample from. This must be followed by a Metropolis Hastings correction step because this TSG does not account for the increments to the elementary tree counts that result from using the same elementary tree more than once in one derivation (represented above by $\mathbb{1}[\cdot]$ function Equation 2.15).

2.4 Summary

In this chapter, we discussed a number of concepts from statistical machine learning as well as grammars and syntax. We emphasized the importance of Bayesian priors and the use of nonparametric modeling techniques in preventing overfitting and underfitting respectively. We introduced a portfolio of grammar formalisms from flat finite-state models to ever richer hierarchical recursive grammars, and illustrated how the Bayesian nonparametric inference techniques can facilitate the induction of such grammars. Next, we will begin presenting our thesis experiments by discussing how web-scale Wikipedia data can help induce models with rich linguistic structure, and lead to qualitative and quantitative gains.

Chapter 3

Power of Web-scale Data: Finer-grained Solutions

What makes a model linguistically rich is the ability to deploy an abundance of parameters that individually explain a large variety of linguistic phenomena, such as idiomatic expressions, lexical dependencies, verb subcategorization, long-range gender or number coordination, and so on. We have argued that, within the probabilistic framework, one cannot simply highly parameterize a model and expect to uncover such linguistic phenomena unless these phenomena are heavily attested in the training data as frequent patterns. In this chapter we demonstrate how large quantities of data, such as found in Wikipedia, facilitate using highly-parameterized models that not only recover rich linguistic structure but also lead to improved predictive performance. We investigate this claim in two separate application domains: extractive sentence compression (Section 3.1) and lexical correction

(Section 3.2).[1]

3.1 Mining Wikipedia revision histories for improving sentence compression

We have introduced sentence compression as an interesting application of machine translation and grammar induction technology in Section 1.1. A well-recognized limitation of research on supervised sentence compression is the dearth of available training data. We propose a new and bountiful resource for such training data, which we obtain by mining the revision history of Wikipedia for sentence compressions and expansions. Using only a fraction of the available Wikipedia data, we have collected a training corpus of over 380,000 sentence pairs, two orders of magnitude larger than the standardly used Ziff-Davis corpus. Using this newfound data, we propose a novel lexicalized noisy channel model for sentence compression, achieving state-of-the-art results when tested out-of-domain, demonstrating the overall generality and robustness of our approach.

3.1.1 Introduction

With the increasing success of machine translation (MT) in recent years, several researchers have suggested transferring similar methods for monolingual text rewriting tasks. In particular, Knight and Marcu (2000) (KM) applied a channel model to the task of *sentence compression* – the task of summarizing a sentence while retaining most of the in-

[1]This chapter is substantially based on previously published papers, with text used by permission of the authors (Nelken and Yamangil, 2008; Yamangil and Nelken, 2008).

formational content and remaining grammatical (Jing, 2000). In *extractive* sentence compression, which they focused on, an order-preserving subset of the words in the sentence are selected to form the summary, that is, we summarize by deleting words. An example sentence pair is the following:

- Like FaceLift, much of ATM's screen performance depends on the underlying application.
- ATM's screen performance depends on the underlying application.

where the underlined words were deleted. In *supervised* sentence compression, the goal is to generalize from a parallel training corpus of sentences (source) and their compressions (target) to unseen sentences in a test set to predict their compressions. An *unsupervised* setup also exists; methods for the unsupervised problem typically rely on language models and linguistic/discourse constraints (Clarke and Lapata, 2006a; Turner and Charniak, 2005).

Compressed sentences can be useful either on their own, for example, as subtitles or captions, or as part of a larger summarization or MT system. A well-recognized problem with this approach, however, is data sparsity. While bilingual parallel corpora are abundantly available, monolingual parallel corpora, and especially collections of sentence compressions are vanishingly rare. Indeed, most work on sentence compression has used the Ziff-Davis corpus (Knight and Marcu, 2000), which consists of a mere 1067 sentence pairs. While data sparsity is a common problem of many NLP tasks, it is much more severe for sentence compression, leading Turner and Charniak (2005) to question the applicability of the channel model for this task altogether.

Our contribution in our sentence compression experiments is twofold. First, we solve the data sparsity issue by showing that abundant sentence compressions can be extracted from Wikipedia's revision history. Second, we use this data to validate the channel model approach for sentence compression, and improve upon it by creating a novel fully lexicalized compression model in which every phrasal constituent is annotated with its head word (Collins, 1997). By doing this, we are able to account for long-range lexical dependencies and better model the context and meaning of a compression rule. We demonstrate that this highly-parameterized model is only affordable under large-scale data conditions, and propose ways of smoothing the model so as to maintain a balance between sophistication and robustness.

3.1.2 Data: Wikipedia revision histories as a source of sentence compressions

Many researchers are increasingly turning to Wikipedia as a large-scale data source for training NLP systems (Strube and Ponzetto, 2006; Cucerzan, 2007; Mihalcea, 2007). The vast majority of this work uses only the most recent version of the articles. In fact, Wikipedia conveniently provides not only the latest version, but the entire revision history of each of its articles, as dramatically visualized by Viégas et al. (2004). Through Wikipedia's collaborative editing process, articles are iteratively amended and refined by multiple Web users. Users can usually change any aspect of the document's structure and content, but for our purposes here, we focus only on sentence-level edits that add or drop words.

We mined a subset of the 1.4 million articles in the July 2006 snapshot of the English

Wikipedia for such compressions/expansions. We make the simplifying assumption that *all* such edits also retain the core meaning of the sentence, and are therefore valid training data for our purposes. This assumption is of course patently naive, as there are many cases in which such revisions reverse sentence meaning, add or drop essential information, are part of a flame war, etc. Classifying these edits is an interesting task which we relegate to future work.[2]

From about one-third of the snapshot, we extracted over 380,000 sentence pairs, which is two orders of magnitude more than the Ziff-Davis corpus. We provide a quick view into this data in Table 3.1.

More technically, for each article, we first extract all revisions, and split each revision into a list of its sentences.[3] We run an edit-distance comparison between each such pair, treating each sentence as an atomic "letter". We look for all replacements of one sentence by another and check whether one sentence is a compression of the other.[4] We then run the syntactic parser of Collins (1997), using just the sentence pairs where a syntactic analysis can be obtained with high confidence (negative log likelihood below 200).

3.1.3 Noisy channel model

We follow KM in modeling the problem using a generative noisy channel model, but use the newfound training data to lexicalize the model. Sentences start their life in short

[2]For instance, compressions are more likely to signal optional information than expansions; the lexical items added are likely to be indicative of the type of edit, etc.

[3]We used a sentence splitter by Paul Clough, from `http://ir.shef.ac.uk/cloughie/software.html`.

[4]We ignore word re-orderings or replacements that are beyond word addition or deletion.

Table 3.1: Example compressions collected from Wikipedia revisions, "b" for before, "a" for after.

b:	From 1926 to 1929 he lived in the British Mandate of Palestine~~, partly in a "kibbutz"~~.
a:	From 1926 to 1929 he lived in the British Mandate of Palestine.
b:	But taken as a whole, his writings are ~~well~~ worth serious consideration.
a:	But taken as a whole, his writings are worth serious consideration.
b:	Capturing their frustrations ~~with this state of affairs~~, Koestler described these people as the "screamers".
a:	Capturing their frustrations, Koestler described these people as the "screamers".
b:	He claimed to have produced ~~possibly~~ the first crosswords in Hebrew.
a:	He claimed to have produced the first crosswords in Hebrew.
b:	In 1983, Koestler, suffering from Parkinson's disease and leukemia, committed joint suicide ~~by taking an overdose of drugs~~ with his third wife Cynthia.
a:	In 1983, Koestler, suffering from Parkinson's disease and leukemia, committed joint suicide with his third wife Cynthia.
b:	Schopenhauer ~~was influenced by Friedrich Schelling,~~ called himself a Kantian, and despised Hegel.
a:	Schopenhauer called himself a Kantian, and despised Hegel.
b:	He died of natural causes on September 21 of the same year ~~at the age of 72~~.
a:	He died of natural causes on September 21 of the same year.
b:	He was ~~arguably~~ one of the most important 19th century philosophers, most famous for his work "The World as Will and Representation".
a:	He was one of the most important 19th century philosophers, most famous for his work "The World as Will and Representation".

form, s, are ranked by a source language model, $p(s)$, and then probabilistically expanded to form the long sentence, $p(l \mid s)$. During decoding, given a long sentence, we seek the most likely short sentence that could have generated it. Using Bayes' rule, this is equivalent to seeking the short sentence s that maximizes $p(l \mid s)p(s)$.

Lexicalized channel model

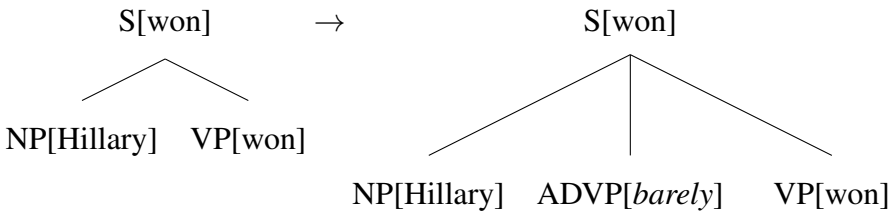
KM's original model was purely syntax-based. Due to the scarcity of available compression data, any modeling efforts had to include harsh levels of abstraction. However, Daumé et al. (2002) used a lexicalized PCFG to *rerank* the compressions, showing that the addition of fine-grained lexical information (the identity of words comprising a constituent) in the source language model helps eliminate improbable compressions and favor plausible ones by modeling lexical correlations that enforce syntactic and semantic coherence. Here, we propose to enhance lexicalization by including lexical information directly within the channel model, allowing us to better model which compressions are likely and which are not. We are able to do this by virtue of our larger dataset, since our model parameters will now represent fine-grained lexical dependencies in addition to syntactic dependencies. A minimal example pair illustrating the utility of lexicalization is the following.

(1) Hillary *barely* won the primaries.

(2) Hillary *almost* won the primaries.

The validity of dropping the adverbial here clearly depends on the lexical value of the adverb. It is more acceptable to drop the adverb in sentence (1), since dropping it in

sentence (2) reverses the meaning. We learn channel probabilities of the form:



where the short rule on the left hand side is expanded into the long rule on the right hand side. Our model has the power of making compression decisions based on lexical dependencies between the compressed and retained parts of the parse tree.

Note that Daumé et al. (2002)'s reranking model cannot achieve this type of distinction, since it is based on reranking the compressed version, at which point the adverb is no longer available.

Since we are interested not only in learning how to compress, but also when to compress, we also include in this procedure unchanged CFG rule pairs that are attested in the corpus. Thus, different ways of expanding a CFG rule compete with each other as well as the possibility of not doing any expansion.

Smoothing

In order to smooth our estimates we use Witten-Bell discounting with six levels of back-off (Witten and Bell, 1991). This method enables us to tune the confidence parameter associated with an estimate inversely proportionally with the diversity of the context of the estimate. The different levels are illustrated in Table 3.2. Level 1, the most specific level, is fully lexicalized. Transitioning to levels 2 to 4, we lose the lexical information about the subtrees that are not dropped, the head child bearing subtree, and the dropped subtrees, respectively. At level 4, we end up with the non-lexicalized estimates that are equivalent

Table 3.2: Back-off levels

level	expanded	short
1	S[won] → NP[Hillary] ADVP[barely] VP[won]	S[won] → NP[Hillary] VP[won]
2	S[won] → NP ADVP[barely] VP[won]	S[won] → NP VP[won]
3	S → NP ADVP[barely] VP	S → NP VP
4	S → NP ADVP VP	S → NP VP
5	parent = S, head-child = VP, child = ADVP	parent = S, head-child = VP
6	parent = S, child = ADVP	parent = S

to KM's model. In subsequent back off levels, we abstract away from the CFG rules. In particular, level 5 estimates the probability of dropping subtrees in the context of a certain parent and head child, and level 6 estimates the probability of the same outcome in the coarser context of a parent only.

Source model

In addition to the lexicalized channel model, we also use a lexicalized probabilistic syntax-based source model, which we train from the parser's output on the short sentences of each pair.

Decoding

We implemented the forest-based statistical sentence generation method of Langkilde (2000). KM tailored this method to sentence compression, compactly encoding all

compressions of a sentence in a forest structure. The forest ranking algorithm, which extracts compressed parse trees, optimized the model scores as well as an additional bigram score. Since our model is lexicalized, the bigram scores become less relevant, which was confirmed by experimentation during development. Therefore in our implementation we exclude the bigram scores and other related aspects of the algorithm such as pruning of bigram-suboptimal phrases.

3.1.4 Evaluation

We evaluated our system using the same method as KM, on the same 32 sentences taken from the Ziff-Davis corpus. We solicited judgments of *importance* (the value of the retained information), and *grammaticality* for our compression, the KM results, and human compressions from 8 judges, on a scale of 1 (worst) to 5 (best). Mean and standard deviation are shown in Table 3.3. Our model improves grammaticality with only a slight decrease in importance at a harsher compression rate. Here are some illustrative examples, with the deleted material shown in brackets:

(3) The chemical etching process [used for glare protection] is effective and will help if your office has the fluorescent-light overkill [that's typical in offices].

(4) Prices range from $5,000 [for a microvax 2000] to $179,000 [for the vax 8000 or higher series].

We suspect that the decrease in importance stems from our indiscriminate usage of compressions and expansions to train our system. We hypothesize that in Wikipedia, expansions often add more useful information, as opposed to compressions which are more

Table 3.3: Evaluation results

	KM	Our model	Humans
Compression	72.91%	67.38%	53.33%
Grammaticality	4.02±1.03	4.31±0.78	4.78±0.17
Importance	3.86±1.09	3.65±1.07	3.90±0.58

likely to drop superfluous or erroneous information.[5] Further work is required to classify sentence modifications.

3.1.5 Discussion

Turner and Charniak (2005) question the viability of a noisy channel model for the sentence compression task. Briefly put, in the typically sparse data setting, there is no way to distinguish between the probability of a sentence as a short sentence and its probability as a regular sentence of English. Furthermore, the channel model is likely to prefer to leave sentences intact, since that is the most prevalent pattern in the training data. Thus, they argue, the channel model is not really compressing, and it is only by virtue of the length penalty that anything gets shortened at all. Our hope here is that by using a far richer source of short sentences, as well as a huge source of compressions, we can overcome this problem. The noisy channel model posits a virtual competition on each word of coming

[5]For instance, here is an expansion seen in the data, where the added information (bracketed) is important: "In 1952 and 1953 he was stationed in Sendai, Japan during the Korean War [and was shot]." It would be undesirable to drop this added phrase.

either from the source model (in which case it is retained in the compression) or from the channel model (in which case it is dropped). By having access to a large data set for the first time, we hope to be able to learn which parts of the sentence are more likely to come from which of the two parts of the model. Further work is required in order to clarify this point.

3.2 Scalable lexical correction from Wikipedia edits using perceptron reranking

In the previous section we used extractive sentence compression to test our hypothesis that the linguistically rich models that are made affordable by the use of web-scale data can bring about qualitative and quantitative gains. By using a fully lexicalized channel model we were able to capture informative compression patterns and make robust estimates for their corresponding probabilities, which in turn gave us a generalizable and robust system that performed well even in out-of-domain settings. In this section we investigate the same hypothesis in an alternative domain and using a different class of models, namely lexical correction and hidden Markov models.

We propose a novel model of large-scale lexical correction of all document words, including both context-sensitive spelling correction and stylistic lexical modifications, trained on Wikipedia's edit revisions. Since Wikipedia articles are edited collaboratively, errors introduced by one writer are likely to be subsequently corrected by others. We mine a set of 1.5 million such correction samples. In our task, we wish to correct all possible errors, rather than focusing on a set of predetermined target words, making the learning problem

much more difficult. We use the abundant Wikipedia data to train a novel model of text correction based on a generative HMM and a reranking perceptron that is used to incorporate fine-grained lexical and semantic evidence towards correction decisions. Once again, such a highly-parametrized generative model can be effectively trained only in the presence of web-scale data. We evaluate our model against context-sensitive spelling correction, obtaining state-of-the-art accuracy at a more general setting.

3.2.1 Introduction

Text is often fraught with errors in spelling, style, and grammar. Traditional spell-checking—the problem of replacing non-lexicon words with their most likely close variant in the dictionary—is an all but trivial engineering task by today's standards.[6] Correcting contextual spelling errors—where the input word is valid but incorrect within its context, such as *peace-piece*—is more challenging. Although modern word processors provide support for text correction, even the most sophisticated tools still fall short of catching all errors.[7]

Given a collection of typical errors and their corrections, supervised learning approaches can help to automatically correct text, possibly more accurately and with broader coverage than either rule-based or unsupervised approaches. For instance, for context-sensitive spelling correction, Carlson et al. (2001) presented a Winnow-based algorithm achieving accuracy levels in the 99% range for 265 such tuples called "confusion sets". Given a hand-

[6] See `http://norvig.com/spell-correct.html` for a bare-bones unigram spelling corrector.

[7] See `http://faculty.washington.edu/sandeep/check/` for a critique of a popular commercial text editor's correction capabilities.

ful of alternatives for a given word, Winnow can learn contextual features to disambiguate between the different alternatives. It is unclear, however, to what extent this approach can scale to not only correcting pre-determined words, but all words in a sentence, and moreover, allowing more than a small set of pre-determined alternatives for each word. Following this approach to its logical conclusion, we would have first to either manually specify or automatically induce confusion sets for every word, collect sufficient training data to distinguish between the alternatives, and finally do feature selection and training separately for each word.

This distinction is analogous to the distinction in the Word Sense Disambiguation literature (Agirre and Edmonds, 2006) between *lexical sample* and *all words* disambiguation. Whereas in lexical sample disambiguation, a system is given a fixed set of words and their senses to disambiguate, the all-words task calls for disambiguation of all the words in a text. This latter task has proven to be much more challenging.

Our contribution here is twofold. First, we present a novel approach to obtaining lexical correction data, by mining Wikipedia to obtain large-scale parallel corpora for text correction (Section 3.2.2). We limit context to a sentence, and frame the text correction task as a word replacement task, ignoring the possibility of correcting a sentence by reordering or replacing words and phrases. Second, we propose a new model for learning to correct text. We first use a baseline hidden Markov model (HMM) trained on the Wikipedia correction edits (Section 3.2.3), and then further augment it with perceptron reranking (Section 3.2.4), adding additional contextual features, including surrounding content words and semantically related words. Despite the better generality of our approach, we evaluate it on the more restricted task of disambiguating between members of confusion sets, achiev-

ing state-of-the-art accuracy (Section 3.2.5). Moreover, we provide examples of the more general corrections that our method is capable of.

3.2.2 Training data: 1.5 million corrections from Wikipedia

Brill and Moore (2000) used a collection of common real-world spelling mistakes to improve context-sensitive spelling correction. Amassing a large-scale collection of spelling errors and their corrections is therefore of obvious interest for improving correction accuracy.

Wikipedia is a promising resource for textual correction as illustrated by our related work (Nelken and Yamangil, 2008) whereby we show that a particular form of lexical correction known as "eggcorns" can be successfully identified within Wikipedia revisions. Eggcorns, first introduced on the popular "Language Log" blog[8] are defined as a replacement of a word by a homophone such that the replacement, while erroneous, still makes some semantic sense. For instance, "eggcorn" is itself an eggcorn for "acorn". Hence, the coining of the term. We were able to find 31% of a reference collection of eggcorns by comparing adjacent revisions of the same article, as well as identify many new previously unidentified eggcorns.

We wish to follow a similar methodology to Section 3.1 to obtain training data for lexical correction from Wikipedia revisions. We extract all sentence replacements; however this time, instead of retaining all sentences that were changed by adding/dropping words we retain all sentence pairs that differ by one word. We make the simplifying assumption that *all* such edits are actually valid lexical corrections, and therefore appropriate training data

[8] `http://itre.cis.upenn.edu/~myl/languagelog/`

for our purposes. This assumption is of course false, since there are many cases in which such revisions change factual information, resolve pronouns, introduce errors, are part of a flame war, include profanity, etc. Clearly, this revision data requires significant clean-up to improve its quality, which is a fascinating classification task that we leave for future work. Using this method, we extracted about 1.5 million sentence pairs.The English section of Wikipedia currently has over 4.2 million articles at the time of writing and is continually expanding both in the number of articles and in the number of revisions. Thus, if we can train a high-precision low-recall classifier to clean up the data, we could significantly improve the quality of the training data, without seriously compromising quantity. We use the Wikipedia data to train an HMM.

3.2.3 HMM for contextual correction

We use an HMM to model the generative process of writing. This model is similar in spirit to that of Mays et al. (1991) for context-sensitive spelling correction, except instead of limiting the allowable corrections to a fixed set of edit operations, we allow any corrections found in the training data, even if the words are phonetically unrelated. Given an input sentence, which we view as a sequence of words, $\mathbf{x} = x_1, x_2, \ldots, x_N$, where one or more words are potentially erroneous, we wish to predict a possibly different sequence of correct words $\mathbf{y} = y_1, y_2, \ldots, y_N$. Since traditional non-lexicon spelling correction is uninteresting, we assume that all words in the input sequence $\mathbf{x}$, are valid words, such that $x, y \in V$: the set of all vocabulary words. Thus in practical usage, we would run our model only after regular non-lexicon spelling errors would be eliminated.

We expand the joint probability $p(\mathbf{x}, \mathbf{y})$ of intending $\mathbf{y}$ but writing $\mathbf{x}$ into a Markov chain

with the following independence structure:

$$p(\mathbf{x},\mathbf{y}) = \prod_{n=1}^{N} p(y_n \mid y_{n-2}, y_{n-1}) \cdot p(x_n \mid y_n)$$

This implies that during writing the intended correct words are chosen conditioned on the previous two words, and for each word, we are allowed to either emit the word itself, or some other word which is either a spelling or a stylistic variant of the original. Thus, the result might potentially be incorrect.

We use the Wikipedia samples to train the HMM as follows. We estimate the transition probabilities from the correct member of each of the training sentence pairs. We train the emission probabilities from the lexical corrections we encounter in the data.

Decoding is the search for the most likely correct sentence $\mathbf{y}^* = \text{argmin}_{\mathbf{y}} - \log p(\mathbf{x},\mathbf{y})$ that could have generated the observed sentence $\mathbf{x}$. Despite the large hidden state space V, the standard Viterbi algorithm can be performed efficiently by storing maps from x to potential corresponding y as constrained by the data.

Smoothing

Storing every word encountered in our training dataset results in a lexicon of 1.5 million entries, causing prohibitive memory requirements on estimating the model parameters. We therefore map any word not in our 50K-word lexicon to a special unknown token. This not only discards spelling errors and their corrections but also gives us statistics over unseen words, effectively smoothing the model. In an attempt to further smooth our estimates, we linearly interpolate different maximum likelihood estimates as in the following.

$$\begin{aligned} p(y_n \mid y_{n-1}, y_{n-2}) &= \lambda_1^1 \cdot p(y_n \mid y_{n-1}, y_{n-2}) + \lambda_2^1 \cdot p(y_n \mid y_{n-1}) + \lambda_3^1 \cdot p(y_n) \\ p(x_n \mid y_n) &= \lambda_1^2 \cdot p(x_n \mid y_n) + \lambda_2^2 \cdot p(x_n) \end{aligned}$$

The interpolation weights were fit to a development set by grid search.

3.2.4 Perceptron reranking

While powerful, the HMM is limited to local trigram information, and thus does not exploit all the useful information in the sentence, both for training and during actual correction. In particular, the content words in the sentence are likely to be a good indicator of correct usage. For instance, consider distinguishing between *ballot-ballet*. Since these words belong to very different contextual domains, we can expect very different words to co-occur within the same sentence. If we encounter "ballet" in a sentence surrounded by words that are relevant mostly to elections, we would like to be able to correct it to "ballot". Hirst and Budanitsky (2005) proposed a context-sensitive spelling corrector based on this idea, by checking semantic relations defined over WordNet (Fellbaum, 1998). We would like to be able to take advantage of such features not as an alternative to the HMM, but as a supplement to it.

One possible solution is to use a sequence-based discriminative model, which can take into account both sequence information and additional features. We experimented with SVM-HMM (Tsochantaridis et al., 2005), but found that it was well-tuned to a limited number of hidden state labels. In our setting, the hidden state space is the 50K vocabulary, causing a performance bottleneck (at least as currently implemented).

We therefore chose instead to apply a discriminative reranking algorithm to the K-best output of the HMM. For a given sentence, we first apply the baseline HMM, and run K-best Viterbi decoding on it (using $K = 25$). We then use a perceptron trained on additional

features to rerank these results (Collins and Koo, 2005).[9] We now explain this process in more detail.

Features

We use the following features to train the perceptron. First, we use the three trigrams focused around the correct target word, y_n, i.e., $\langle y_{n-2}, y_{n-1}, y_n \rangle$, $\langle y_{n-1}, y_n, y_{n+1} \rangle$, and $\langle y_n, y_{n+1}, y_{n+2} \rangle$. Note that this and all other features are encoded as binary indicator features. Thus, for every possible trigram t in the training data, we have a feature of the form:

$$f_t = \begin{cases} 1 & \text{if } t \text{ in window around } y_n \\ 0 & \text{otherwise} \end{cases}$$

To add content words, we define content words to simply be any word not contained in a stopword list. We search for the first content word on the left of the target word, x_l, and the first content word on the right, x_r, adding features encoding the pairs (x_l, y_n), and (y_n, x_r).

In addition to all content words, we put in another feature encoding the first semantically related word on the left (skipping any stopwords in-between) and on the right using the JCN relation (Jiang and Conrath, 1998), which Budanitsky and Hirst (2006) found to be the most effective one for this task. JCN is defined on nodes—sets of noun synonyms

[9] Using the perceptron alone on a per-confusion-set basis is also a valid option. Although we did not conduct such as experiment, we would expect the results to be similar to the Winnow approach of Carlson et al. (2001), which we compare our results to. The benefit of our reranking approach is being able to combine the sequence tagging power of the generative HMM with the contextual modeling power of the discriminative perceptron.

(synsets)—in the WordNet hierarchy, and is computed as follows:

$$\text{JCN}(s_1, s_2) = \quad -\log p(s_1) - \log p(s_2) + 2\log \text{LCS}(s_1, s_2)$$

where $\text{LCS}(s_1, s_2)$ is the lowest common subsumer of s_1 and s_2 in the WordNet is-a hierarchy, and probabilities $p(s_1)$ are defined as the probability of encountering one of the members of the synset s_1 or any of its hyponyms in a corpus. Specifically, they obtain these probabilities from the Brown corpus. JCN is extended from synsets to words by taking the minimum over all possible senses of the two words:

$$\text{JCN}(w_1, w_2) = min_{s_1, s_2} \text{JCN}(s_1, s_2) \tag{3.1}$$

where $w_1 \in s_1, w_2 \in s_2$.

As Budanitsky and Hirst (2006) show, this measure correlates well with human judgments of word similarity (Rubenstein and Goodenough, 1965).[10] They use this correlation to determine a threshold for binarizing the relation, determining whether a pair of words are related or not.[11]

Pedersen et al. (2004) provide an open-source Perl package to compute JCN for any word pair, by traversing the WordNet hierarchy at runtime. For use in training, we wished to optimize this computation by pre-computing the entire relation up to the threshold. Such pre-computation is impractical with the current package, which takes on average about 65

[10] Hirst and Budanitsky report a correlation of -0.781. Using a newer version of WordNet (2.0 vs. 1.5) we got an even better correlation of -0.8535.

[11] To replicate their choice of a threshold, we define $\text{JCN}'(w_1, w_2) = \mathbb{1}[\text{JCN}(w_1, w_2) > th]$. Choosing any value between 5.5 and 7.5 for th yields an optimal correlation of -0.8544, slightly better than the original correlation value without a threshold. For concreteness, to obtain maximal coverage, we chose $th = 7.5$.

milliseconds to compute the relation for a pair of nouns. Given that we need to do this for many tens of thousands of synonym pairs, pre-computing the entire relation would take on the order of many months.

Instead, we precomputed the relation for the entire WordNet hierarchy using the recursive top-down traversal algorithm described in Algorithm 1, which we apply to each of the roots of the WordNet hierarchy. The entire computation was completed in under 19 hours on stock hardware. This algorithm computes the relation on synsets. To apply to words, we compute the symmetric closure and then apply the minimum operation from Equation 3.1.

Running the perceptron

We train the perceptron on $(\mathbf{x}, \mathbf{y})$ pairs, where $\mathbf{x}$ is the input sequence, and $\mathbf{y}$ is a correction, as well as a designated target word that differs between them, which we denote by x_t, and y_t, respectively. For simplicity we assume only one correction per sentence. We obtain training instances from the same Wikipedia sentence pairs we used to train the HMM.

In addition to the Wikipedia data, we use single sentences from raw text corpora, which we randomly perturb by introducing a deliberate error in one of the words. This is useful to specifically train the model for specific types of errors, as we discuss in more detail in Section 3.2.5.

We train the perceptron as in Algorithm 2, where W is a vector of weights, one per feature. We initialize the weights to zero and iterate this training procedure 10 times, a number we optimized on held out training data.

During training, for each input sentence, and its top-K corrections generated by the HMM, we use the current weights to calculate the inner products with the corresponding

feature representation and choose the correction maximizing this value as our prediction. If the prediction does not agree with the true correction of the sentence, we perform a linear update of the weight vector.

Trained weights are finally used to define a "reranking" of the top-K corrections of a test sentence. Typically, we predict the one that maximizes the same inner product value.

3.2.5 Evaluation

Evaluation on context-sensitive spelling

Although our system is more general than standard context-sensitive spelling correction, for easy comparison with earlier work, we evaluated it on 12 binary confusion sets. To increase the number of training instances that are relevant to the confusion sets, we augmented the Wikipedia training samples with sample sentences containing members of these confusion sets from additional raw text one-sided corpora. We use the Brown corpus, the Penn Treebank, and two additional open-source textual collections included in the NLTK distribution (Loper and Bird, 2002): presidential inaugural addresses, and Australian broadcast news. From one-sided raw training sentences, which we assume are all valid, we generate an incorrect sentence by randomly flipping a confusion set member to the other one. We used an additional 16K sentences from these corpora for training, and 5K more for testing.

Results are shown in Table 3.4 and Table 3.5. We compare our results with those of Carlson et al. (2001). Since their system does not always make a prediction, they define the following notions for evaluation. *Performance* is the percentage of the predictions the system makes that are correct, and *willingness*, the percentage of queries (occurrences of

confusion set members) on which the system makes a prediction. For our system, we also provide precision, recall, and $f1$-value. Our willingness is effectively always 1, since we are always ready to make a prediction. Since our precision is very high, there is no need to reduce willingness.

Interestingly, the results were exactly the same whether we used the features for all left and right content words, just the JCN-related content words, or both. This would seem to indicate that the semantically related content words contain the same information as the full set of content words at least for these confusion sets, and when restricted to just one word to the left and one word to the right. Thus, to the extent that adding content words helps the classification, on one hand apparently only the semantically related words are useful. On the other hand, apparently we do not need to determine the set of words in advance, and can just let the perceptron generalize by itself which words are relevant and which are not. Of course, more extensive experiments are required in order to see whether this property holds more generally.

Beyond confusion sets

It is important to note that although the evaluation above is on restricted confusion sets, we did not restrict the output of our system to the members of the binary confusion sets. Thus, our system is more general, and can handle open-ended corrections. We illustrate this characteristic using some example erroneous corrections made by our system. For each example, the first sentence is the gold standard (g), the second is the test sentence (t), and the third one gives our model's prediction (p). We count these as errors of course, even though the corrections are arguably just as good as the gold version.

1. [g:] They might benefit from *their* treatment there.

   [t:] They might benefit from *there* treatment there.

   [p:] They might benefit from *the* treatment there.

2. [g:] However, my *principal* objection in this sort of novel is to the hackneyed treatment of (. . .)

   [t:] However, my *principal* objection in this sort of novel is to the hackneyed treatment of (. . .)

   [p:] However, my *main* objection in this sort of novel is to the hackneyed treatment of (. . .)

3. [g:] Living pictures of the early boroughs, *country* life in Tudor and Stuart times, the impact of (. . .)

   [t:] Living pictures of the early boroughs, *country* life in Tudor and Stuart times, the impact of (. . .)

   [p:] Living pictures of the early boroughs, *rural* life in Tudor and Stuart times, the impact of (. . .)

By eliminating the dependence on confusion sets, we allow more general corrections. To illustrate this, we ran the system on the following actual errors we happened to find in the text of the Carlson et al. (2001) paper, where the first sentence of each example is the erroneous sentence and the second one is our system's output.

1. (a) This work makes *used* of the concept of confusion sets (. . .)

   (b) This work makes *use* of the concept of confusion sets (. . .)

2. (a) All of the experiments were performed using (. . .) and *a* initial weight of 0.2.

(b) All of the experiments were performed using (. . .) and *an* initial weight of 0.2.

3. (a) Overall effects of eligibility on all our *confusions* sets as well as (. . .)

   (b) Overall effects of eligibility on all our *confusion* sets as well as (. . .)

It should be noted, however, that by using the additional training data for each member of the confusion sets, we are essentially training the perceptron more specifically on these confusion sets. Thus, while the model is theoretically unaware of confusion sets in any explicit sense, by this special additional training, it is essentially being fine-tuned to them. Thus, to get the maximum potential accuracy from the model, we are back to the problem of having to obtain specific training data for each confusion set.

The advantage of our model is that it is able to combine the best of both worlds. It is not only able to do general correction, but within the same framework can also be further trained for specific confusion sets. What is admittedly lacking from our evaluation is a further evaluation of the correction accuracy of the model on general texts, and a comparison of the baseline and reranked model on such texts. We leave this further evaluation for future work.

3.2.6 Discussion

We presented a novel model of general lexical corrections using an HMM trained on correction data extracted from Wikipedia, which we then rerank using a perceptron. Our model combines several advantages. First, it is general in that it can be applied to texts without requiring a notion of confusion sets. It thus combines the generality of the baseline HMM with more advanced contextual features. Second, by training the model on Wikipedia correction data, the model is attuned to actual corrections made by speakers of

English. Third, in an evaluation on the more specific task of contextual spelling correction on confusion sets, our model achieves state-of-the-art accuracy levels, given additional single-sided training sentences per confusion-set member.

The lexical and semantic information encoded by the contextual features of the perceptron is key to the success of our approach. Therefore, it is crucial to note the role of web-scale data in providing robust statistics to estimate such fine-grained information. In line with our sentence compression experiments of the previous section, our finding here supports our overall hypothesis that large amounts of data can not only boost performance of simpler models, but can also make affordable more sophisticated models.

```
JCN(x, y, LCS) :
    j = −log p(x) − log p(y) + 2 log LCS(x, y)
    if j < th then
        output j
    end

traverse(s1) :
    S ← subtrees(s1)
    while not empty(S) do
        A ← pop(S)
        foreach node s2 of A do
            JCN(s2, s1, s1)
            foreach B in S do
                foreach node s3 of B do
                    JCN(s2, s3, s1)
                end
            end
        end
        traverse(A)
    end
```

Algorithm 1: Efficient computation of all-pairs JCN relation

train(Dataset D, Weights W):

foreach $(\mathbf{x}, \mathbf{y}) \in D$ **do**

Run $\mathbf{x}$ through HMM

$\mathbf{y}_1, \ldots, \mathbf{y}_K \leftarrow$ HMM's Viterbi top-K

$k = \text{argmax}_{k=1,\ldots,K} W \cdot f(\mathbf{x}, \mathbf{y}_k)$

if $\mathbf{y}_k \neq \mathbf{y}$ **then**

$W = W + f(\mathbf{x}, \mathbf{y}) - f(\mathbf{x}, \mathbf{y}_k)$

end

end

Algorithm 2: Perceptron reranking. For each $(\mathbf{x}, \mathbf{y})$, we retrieve HMM's top-K candidates $\mathbf{y}_1, \ldots, \mathbf{y}_K$, and score each one according to the perceptron. If the highest scoring $\mathbf{y}_k$ is different from the gold standard $\mathbf{y}$, we update our perceptron weights so as to demote $\mathbf{y}_k$ and promote $\mathbf{y}$, otherwise continue without any updates.

Table 3.4: Precision and Recall for the baseline HMM and the reranked model for 12 confusion sets

	Baseline HMM			Reranked HMM		
Confusion Set	Precision	Recall	F-measure	Precision	Recall	F-measure
accept-except	100.00	49.80	66.42	100.00	71.56	83.42
affect-effect	99.94	87.25	92.84	99.63	86.58	92.93
among-between	100.00	21.92	35.39	98.49	52.50	68.72
amount-number	87.39	17.82	28.65	88.31	61.31	72.84
fewer-less	89.43	27.53	42.90	93.09	64.32	76.77
I-me	99.07	23.12	38.43	98.51	78.19	87.62
passed-past	100.00	63.82	77.17	99.96	74.97	85.97
peace-piece	95.08	31.65	46.85	100.00	58.52	73.75
principal-principle	97.01	67.76	79.59	97.97	86.43	91.85
raise-rise	96.78	26.84	41.06	97.70	65.16	78.12
than-then	99.77	75.10	86.92	96.48	83.21	89.48
weather-whether	99.86	70.71	82.44	97.54	78.46	86.41
average	97.03	46.94	59.89	**97.31**	**71.77**	**82.32**

Table 3.5: Performance and Willingness for the reranked HMM and the results of Carlson et al. (2001)

	Reranked HMM		Carlson et al. (2001)	
Confusion Set	Performance	Willingness	Performance	Willingness
accept-except	0.86	1.00	0.99	0.80
affect-effect	0.93	1.00	0.97	0.75
among-between	0.78	1.00	0.98	0.63
amount-number	0.78	1.00	0.98	0.63
fewer-less	0.79	1.00	0.98	0.73
I-me	0.92	1.00	0.99	0.95
passed-past	0.87	1.00	0.99	0.80
peace-piece	0.77	1.00	0.99	0.82
principal-principle	0.91	1.00	0.96	0.51
raise-rise	0.81	1.00	0.98	0.72
than-then	0.90	1.00	0.99	0.88
weather-whether	0.87	1.00	0.99	0.91
average	0.85	1.00	0.98	0.76

Chapter 4

Effective Inference for Increased Linguistic Expressivity

We have seen the power of fine-grained information in modeling natural language, in terms of direct lexicalization as well as incorporation of lexical and semantic contextual features. Although large-scale data helped us incorporate this level of information in the first place, our approaches suffered from lack of adaptivity of their level of granularity. In the sentence compression experiments of Section 3.1 we assumed an excessively parameterized model and had to make considerable smoothing efforts to bring this model down to a reasonable level, even in the face of large-scale data. In the lexical correction experiments of Section 3.2 instead of using a single refined HMM, we had to resort to the two-stage method of using a coarse HMM and a separately trained fine-grained perceptron. In neither of these experiments was the level of granularity of our generative models determined by maximizing a meaningful, principled objective.

In this chapter we answer the natural question of how to *automatically* determine the amount and nature of the particular fine-grained information in SGI, especially when the decision is subject to constraints of limited computational resources and particularities of data. We show that the *Bayesian nonparametric* machine learning paradigm that was introduced in Section 2.3.2 can be used to determine the optimal level of granularity in an efficient and fully data-driven way. Nonparametric models fit their number of parameters to the specifics of the data, and therefore fight data sparsity, and prevent over- and underfitting. For example, the existence of lexical items in a grammar rule can be determined automatically during training.

In particular, we build Bayesian nonparametric inference schemes for two linguistically motivated grammar formalisms: TIG and TAG (see Section 2.2 for an introduction), and compare with the already existing schemes for TSG (Cohn and Blunsom, 2010). We use the domain of syntactic parsing on the English treebank as our test-bed (Marcus et al., 1993). Since this is a more abstract domain than the application domains that we investigated in Chapter 3, such as sentence compression and lexical correction, we are able to study the performance and effects of our approach more clearly. We show that nonparametric TIG is a better model for English treebank data than TSG in terms of parsing accuracy, and it achieves this at negligible computational overhead and while providing richer syntactic analyses. Our nonparametric TAG experiments show that while TIG already encapsulates most of the adjunction power needed for modeling English, a number of parenthetical expressions (such as quotations and parentheses) can be captured more accurately with TAG. More importantly, for the first time we provide an efficient (quadratic-time training and cubic-time testing) framework for *unconstrained* TAG which we expect to open new av-

enues for TAG experiments for domains and languages other than English treebank parsing.

These experiments with nonparametric grammar induction serve our overarching goal of using web-scale data to recover rich linguistic structure, as it would be computationally infeasible to fit such flexible grammars to our large amounts of data without a straightforward way of adapting their level of granularity. In the chapter that follows, we will test our nonparametric grammar induction approach in the application domain of sentence compression, and show the true power of web-scale data in recovering valuable linguistic information.[1]

4.1 Estimating compact yet rich tree-insertion grammars

We present a Bayesian nonparametric model for estimating tree-insertion grammars (TIG), building upon recent work on Bayesian inference of tree-substitution grammars (TSG) via Dirichlet processes. Under our general variant of TIG, grammars are estimated via the Metropolis-Hastings algorithm that uses a context-free grammar transformation as a proposal, which allows for cubic-time string parsing as well as tree-wide joint sampling of derivations in the spirit of Cohn and Blunsom (2010). We use the Penn treebank for our experiments and find that our proposal Bayesian TIG model not only has competitive parsing performance but also finds compact yet linguistically rich TIG representations of the data.

[1]This chapter is substantially based on previously published papers, with text used by permission of the authors (Yamangil and Shieber, 2012, 2013).

4.1.1 Introduction

There is a deep tension in statistical modeling of grammatical structure between providing good expressivity — to allow accurate modeling of the data with sparse grammars — and low complexity — making induction of the grammars (say, from a treebank) and parsing of novel sentences computationally practical. At the most trivial end of the expressivity scale, linear grammars (regular models) are low in complexity but so low in expressivity that they cannot express nontrivial structure at all. Context-free grammars (CFG) can express hierarchical structure as manifest in a treebank, and in fact are trivial to use for parameter estimation, at the cost of a barely practical $O(n^3)$ parsing relative to sentence length. But even CFGs (by the very definition of their context-freeness) cannot express direct relationships outside of a parent and immediate children, and unsurprisingly, such models perform poorly in analyzing unseen data. In principle, by expanding the domain of locality, we can achieve better expressivity, and the ability to model more contextual dependencies; the payoff would be better modeling of the data or smaller (sparser) models or both. For instance, constructions that go across levels, like the predicate-argument structure of a verb and its arguments can be modeled by tree-substitution grammars (TSG), with their larger domain of locality (Goodman, 2002).

Recent work that has incorporated Dirichlet process (DP) nonparametric models into TSGs has provided an efficient solution to the daunting model selection problem of segmenting training data trees into appropriate elementary parse tree fragments to form the grammar (Cohn et al., 2009; Cohn and Blunsom, 2010; Post and Gildea, 2009). DP inference tackles this problem by exploring the space of all possible segmentations of the data, in search for fragments that are on the one hand large enough so that they incorporate the

useful dependencies, and on the other small enough so that they recur and have a chance to be useful in analyzing unseen data. Because of their better expressivity, TSGs induced in this way provide better models with sparser grammars.

The elementary trees combined in a TSG are, intuitively, primitives of the language, yet certain linguistic phenomena (notably various forms of modification) "split them up", preventing their reuse, leading to less sparse grammars than might be ideal (Chiang, 2000; Resnik, 1992). For instance, imagine modeling the following set of structures:

- [$_{NP}$ the [$_{NN}$ [$_{NN}$ [$_{NN}$ president] of the university] who resigned yesterday]]
- [$_{NP}$ the [$_{NN}$ former [$_{NN}$ [$_{NN}$ president] of the university]]]
- [$_{NP}$ the [$_{NN}$ [$_{NN}$ president] who resigned yesterday]]

A natural recurring structure here would be the structure "[$_{NP}$ the [$_{NN}$ president]]", yet it occurs not at all in the data. A TAG grammar that hypothesizes auxiliary trees corresponding to adjoining "[$_{NN}$ former *NN*]", "[$_{NN}$ *NN* of the university]", and "[$_{NN}$ *NN* who resigned yesterday]" would be able to reuse the basic structure "[$_{NP}$ the [$_{NN}$ president]]". Unfortunately, TAG's expressivity comes at the cost of greatly increased complexity. Parsing complexity for unconstrained TAG scales as $O(n^6)$, impractical as compared to CFG and TSG's $O(n^3)$; moreover, the model selection problem for TAG is significantly more complicated than for TSG since one must reason about many more combinatorial options with two types of derivation operators. This has led researchers to resort to manual (Doran et al., 1997) as well as heuristic techniques; for example, one can consider "outsourcing" the auxiliary trees (Shieber, 2007), use template rules and a very small number of grammar categories (Hwa, 1998), or rely on head-words and force lexicalization in order to constrain

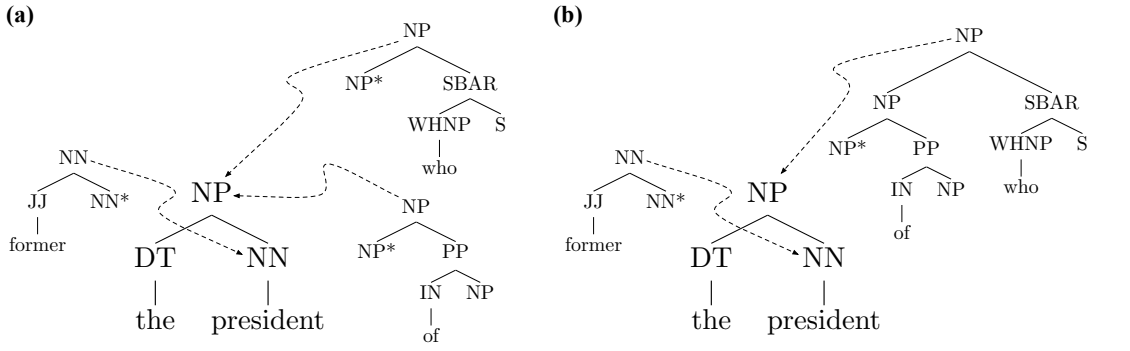


Figure 4.1: Two different TIG derivations of an NP constituent. (a) One left insertion (at NN) and two simultaneous right insertions (at NP) (b) One left insertion (at NN) and one right insertion (at NP) of a depth=2 auxiliary tree.

the problem (Xia et al., 2001; Chiang, 2000; Carreras et al., 2008). Rather than compromise in this way, we would like to do full treebank parsing with the original set of grammar categories, using the nonparametric solution to learn an optimal set of initial and auxiliary trees from the treebank data.

Schabes and Waters (1995) first proposed to use tree-insertion grammars (TIG), a kind of expressive compromise between TSG and TAG, as a substrate on which to build grammatical inference. TIG constrains the adjunction operation so that spliced-in material falls completely to the left or completely to the right of the splice point. By restricting the form of possible auxiliary trees to only *left* or *right* auxiliary trees in this way, TIG remains within the realm of context-free formalisms (with cubic complexity) while still modeling rich linguistic phenomena (Schabes and Waters, 1995). Figure 4.1 depicts some examples of TIG derivations.

Sharing the same intuitions, Shindo et al. (2011) have provided a previous attempt at combining TIG and Bayesian nonparametric principles, albeit with severe limitations.

Their TIG variant (which we will refer to as TIG_0) is highly constrained in the following ways.

1. The foot node in an auxiliary tree must be the immediate child of the root node.
2. Only one adjunction can occur at a given node.
3. Even modeling multiple adjunction with root adjunction is disallowed. There is thus no recursion possibility with adjunction, no stacking of auxiliary trees.
4. As a consequence of the prior two constraints, no adjunction along the spines of auxiliary trees is allowed.
5. As a consequence of the first constraint, all nonterminals along the spine of an auxiliary tree are identical.

In this thesis we explore a Bayesian nonparametric model for estimating fully unconstrained TIG, and compare its performance against TSG and the restricted TIG_0 variant. Our general formulation avoids the TIG_0 limitations.[2] By staging our development incrementally, we show the benefits gained by each feature we support. We test the performance of the following decreasingly constrained models each one adding a new feature to the previous.

- Auxiliary trees may have the foot node at depth greater than one.[3] Both left and right adjunctions (but only one of each) may occur at the same node.

[2]To maintain the TIG properties, we still respect the constraint that only left auxiliary trees can adjoin along the spine of a left auxiliary tree, and symmetrically for right auxiliary trees.

[3]Throughout the thesis, we will refer to the depth of an auxiliary tree to indicate the length of its spine.

- Simultanous adjunction (that is, more than one left or right adjunction per node) is allowed.

- Adjunctions may occur along the spines of auxiliary trees.

- Labels along the spines do not have to match.

The increased expressivity is motivated both linguistically and practically. From a linguistic point of view: Deeper auxiliary trees can help model large patterns of insertion and potential correlations between lexical items that extend over multiple levels of tree. Combining left and right auxiliary trees can help model modifiers of the same node from left and right (combination of adjectives and relative clauses for instance). Simultaneous insertion allows us to deal with multiple independent modifiers for the same constituent (for example, a series of adjectives). The extension is conservative: since our general formalisms are supersets of TIG_0, if it is the case that such a constrained grammar is preferable, our nonparametric inference would have the option to land on such settings. From a practical point of view, we show that an induced TIG provides modeling performance superior to TSG and comparable with TIG_0. However, we show that the grammars we induce are *compact yet rich*, in that they succinctly represent complex linguistic structures measurably more so than TIG_0. We describe how we achieve these generalities flexibly via a grammar transform, and at no significant additional computational overhead.

4.1.2 Probabilistic model

In the basic nonparametric TSG model that was introduced in Section 2.3.2, there is an independent DP for every nonterminal (such as $c = \text{NP}$), each of which uses a base

distribution P_0 that generates an initial tree by making stepwise decisions.

$$G_c \sim \mathrm{DP}(\alpha_c, P_0(\cdot \mid c)) \tag{4.1}$$

The canonical P_0 uses a probabilistic CFG $\tilde{P}$ that is fixed a priori to sample CFG rules top-down and Bernoulli variables for determining where substitutions should occur (Cohn et al., 2009; Cohn and Blunsom, 2010). A derivation is then sampled by getting iid initial trees from G_c and carrying out substitutions, which determines the parse tree. Integrating out G_c we have the exchangeable process of Equation 2.12 for generating initial trees:

$$p(e_i \mid \mathbf{e}_{<i}) = \frac{n_{e_i} + \alpha_c P_0(e_i \mid c)}{i - 1 + \alpha_c} \tag{4.2}$$

where e_i is an initial tree of type c, and n_{e_i} is the number of occurrences of e_i in the previous samples $\mathbf{e}_{<i} = \{e_1, \ldots, e_{i-1}\}$ of the same c type.

We extend this model by adding specialized DPs for left and right auxiliary trees.[4]

$$G_c^{\text{right}} \sim \mathrm{DP}(\alpha_c^{\text{right}}, P_0^{\text{right}}(\cdot \mid c)) \tag{4.3}$$

Therefore, we have an exchangeable process for generating right auxiliary trees

$$p(a_j \mid \mathbf{a}_{<j}) = \frac{n_{a_j} + \alpha_c^{\text{right}} P_0^{\text{right}}(a_j \mid c)}{j - 1 + \alpha_c^{\text{right}}} \tag{4.4}$$

as for initial trees in TSG.

We must define three distinct base distributions for initial trees, left auxiliary trees, and right auxiliary trees. P_0 generates an initial tree with root label c by sampling CFG rules from $\tilde{P}$ and making a binary decision at every node generated whether to leave it as a frontier node or further expand (with probability β_c) following the treatment of Cohn

[4]We use right insertions for illustration; the symmetric analog applies to left insertions.

et al. (2009). Similarly, our P_0^{right} generates a right auxiliary tree with root label c by first making a binary decision whether to generate an immediate foot or not (with probability γ_c^{right}), and then sampling an appropriate CFG rule from $\tilde{P}$. For the right child, we sample an initial tree from P_0. For the left child, if a decision to generate an immediate foot was made, we generate a foot node and stop. Otherwise, we recur into P_0^{right}, which generates a right auxiliary tree that becomes the left child.[5] We use the symmetric process for P_0^{left}.

Table 4.1: Initial tree and auxiliary trees used for illustration of the computation of our base distributions.

ν	:	(SBAR (WHNP who) S)
η_1	:	(NP NP* (PP (IN of) NP))
η_2	:	(NP (NP NP* (PP (IN of) NP)) (SBAR (WHNP who) S))

To illustrate, the initial tree and two right auxiliary trees of Table 4.1 have probabilities defined recursively as follows

$$\begin{aligned}
P_0(\nu) &= \beta_{\text{WHNP}} \times (1-\beta_{\text{S}}) \times \tilde{P}(\text{SBAR} \rightarrow \text{WHNP S}) \times \tilde{P}(\text{WHNP} \rightarrow \text{who}) \\
P_0^{\text{right}}(\eta_1) &= (1-\gamma_{\text{NP}}^{\text{right}}) \times \beta_{\text{PP}} \times \beta_{\text{IN}} \times (1-\beta_{\text{NP}}) \times \tilde{P}(\text{NP} \rightarrow \text{NP PP} \mid \text{NP} \rightarrow \text{NP _}) \\
&\quad \times \tilde{P}(\text{PP} \rightarrow \text{IN NP}) \times \tilde{P}(\text{IN} \rightarrow \text{of}) \\
P_0^{\text{right}}(\eta_2) &= \gamma_{\text{NP}}^{\text{right}} \times \beta_{\text{SBAR}} \times P_0^{\text{right}}(\eta_1) \times P_0(\nu)
\end{aligned}$$

We bring together these three sets of processes via a set of insertion parameters μ_c^{left}, μ_c^{right}. In any derivation, for every initial tree node labeled c (except for frontier nodes) we

[5] With left (right) auxiliary trees, we are guaranteed that spines are along the right (left) hand side edge of the tree, and the foot node should reside at the right (left) hand side corner, as required by the definition of Schabes and Waters (1995).

determine whether or not there are insertions at this node by sampling a Bernoulli(μ_c^{left}) distributed left insertion variable and a Bernoulli(μ_c^{right}) distributed right insertion variable (as in Equation 2.11 but instead of the node itself we use the label of the node to fight data sparsity). For left auxiliary trees, we treat the nodes that are *not* along the spine of the auxiliary tree the same way we treat initial tree nodes; however, for nodes that are along the spine (including root nodes, excluding foot nodes) we consider only left insertions by sampling the left insertion variable (symmetrically for right insertions).

For all types of nodes, we achieve simultaneous insertion by allowing root insertions to occur at the roots of the inserted auxiliary trees. This means that the number of simultaneous left and right insertions is a Geometric(μ_c^{left}) and Geometric(μ_c^{right}) variable respectively, i.e., two left insertions would have probability $(\mu_c^{\text{left}})^2(1-\mu_c^{\text{left}})$.

4.1.3 Inference

Given this model, our inference task is to explore potential derivations underlying the data. Since TIG derivations are highly structured objects, a basic sampling strategy based on local node-level moves such as Gibbs sampling (Geman and Geman, 1984) would not hold much promise. Following previous work, we design a blocked Metropolis-Hastings sampler that samples derivations per entire parse trees all at once in a joint fashion (Cohn and Blunsom, 2010; Shindo et al., 2011). This is achieved by proposing derivations from an approximating distribution and stochastically correcting via accept/reject to achieve convergence into the correct posterior (Johnson et al., 2007). Our approximating distribution is a CFG that approximates the infinite TIG implied by the current state of the DP model, which is discussed next.

4.1.4 Grammar transform

Since our base distributions factorize over levels of tree, CFG is the most convenient choice for a proposal distribution. Fortunately, Schabes and Waters (1995) provide an (exact) transformation from a fully general TIG into a TSG that generates the same string languages. It is then straightforward to represent this TSG as a CFG using the Goodman transform (Goodman, 2002; Cohn and Blunsom, 2010). During sampling, these transformations can be used to represent the finite number of elementary trees previously sampled. The infinite part of the posterior stems from having the infinite base distributions, which we represent by augmenting the transformed CFG with additional rules that represent these sampling options. This way, the proposal distribution has exactly the same support as the full posterior. Figure 4.3 lists the additional CFG productions we have designed, as well as the rules used that trigger them. This overall transformation, although infinite, is still inexact as it disregards the potential increments to the number of occurrences of elementary trees within the derivation of a single tree (in Equations 4.2 and 4.4), which necessitates the Metropolis-Hastings step. See Cohn and Blunsom (2010) for more details.

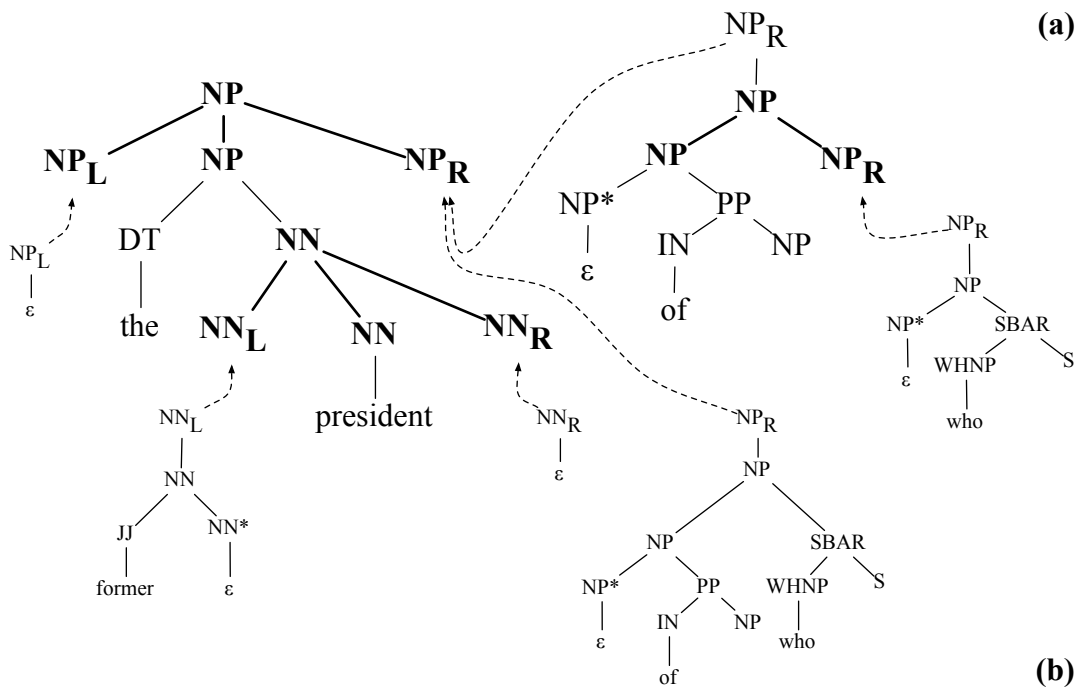


Figure 4.2: **TIG-TSG transform:** (a) and (b) illustrate transformed TSG derivations for two different TIG derivations of the same parse tree structure. The TIG nodes where we illustrate the transformation are in bold. (We suppress the rest of the transformational nodes.) TIG-TSG transform introduces for initial trees a left and a right insertion node (e.g., NP_L, NP_R in the figure) both of which can be expanded via TSG rules to support insertions or can expand directly into the empty string (no insertion). Whereas for nodes along the spines of auxiliary trees, TIG-TSG transform introduces a single left (right) insertion node for left (right) auxiliary trees (e.g., NP_R for the PP auxiliary tree) which can expand in a similar fashion. In the transformed TSG, foot nodes produce only empty strings. Note that we simulate simultaneous insertion via root adjunction between auxiliary trees (the SBAR auxiliary tree being inserted into the root of the PP auxiliary tree).

Table 4.2: Transformation CFG rules that represent infinite base distributions. P_0 is taken from Cohn and Blunsom (2010). Underscored labels (such as $\_\text{NP}^{\text{right}}$ as opposed to NP^{right}) are used to differentiate the pre-insertion nodes in Figure 4.2 from the post-insertion ones. $n_{\text{NP}}^{\text{init}}$ and $n_{\text{NP}}^{\text{right}}$ denote the number of initial and right auxiliary trees of NP type, respectively. P_0^{left} rules are omitted for brevity and mirror the P_0^{right} rules above.

CFG rule	CFG probability
	Base distribution: P_0
$\text{NP} \rightarrow \text{NP}$	$\alpha_c / (n_{\text{NP}}^{\text{init}} + \alpha_c)$
$\text{NP} \rightarrow \text{NP}_{\text{L}}\ \_\text{NP}\ \text{NP}_{\text{R}}$	1.0
$\_\text{NP} \rightarrow \text{DT NN}$	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times (1-\beta_{\text{DT}}) \times (1-\beta_{\text{NN}})$
$\_\text{NP} \rightarrow \text{DT NN}$	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times (1-\beta_{\text{DT}}) \times \beta_{\text{NN}}$
$\_\text{NP} \rightarrow \text{DT NN}$	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times \beta_{\text{DT}} \times (1-\beta_{\text{NN}})$
$\_\text{NP} \rightarrow \text{DT NN}$	$\tilde{P}(\text{NP} \rightarrow \text{DT NN}) \times \beta_{\text{DT}} \times \beta_{\text{NN}}$
	Base distribution: P_0^{right}
$\text{NP}_{\text{R}} \rightarrow \text{NP}^{\text{right}}$	$\mu_{\text{NP}}^{\text{right}} \times \left(\alpha_c^{\text{right}} / (n_{\text{NP}}^{\text{right}} + \alpha_c^{\text{right}}) \right)$
$\text{NP}_{\text{R}} \rightarrow \varepsilon$	$1 - \mu_{\text{NP}}^{\text{right}}$
$\text{NP}^{\text{right}} \rightarrow \_\text{NP}^{\text{right}}\ \text{NP}_{\text{R}}$	1.0
$\_\text{NP}^{\text{right}} \rightarrow \text{NP}_*\ \text{SBAR}$	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP \_}) \times (1-\gamma_{\text{NP}}^{\text{right}}) \times (1-\beta_{\text{SBAR}})$
$\_\text{NP}^{\text{right}} \rightarrow \text{NP}_*\ \text{SBAR}$	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP \_}) \times (1-\gamma_{\text{NP}}^{\text{right}}) \times \beta_{\text{SBAR}}$
$\_\text{NP}^{\text{right}} \rightarrow \text{NP}^{\text{right}}\ \text{SBAR}$	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP \_}) \times \gamma_{\text{NP}}^{\text{right}} \times (1-\beta_{\text{SBAR}})$
$\_\text{NP}^{\text{right}} \rightarrow \text{NP}^{\text{right}}\ \text{SBAR}$	$\tilde{P}(\text{NP} \rightarrow \text{NP SBAR} \mid \text{NP} \rightarrow \text{NP \_}) \times \gamma_{\text{NP}}^{\text{right}} \times \beta_{\text{SBAR}}$

We use a variation of the TIG-TSG transform that is slightly different from the one proposed by Schabes and Waters (1995). Instead of having an explicit simultaneous insertion mechanism and disallowing insertions at the roots of auxiliary trees, we keep the auxiliary tree root insertions and use those to simulate simultaneous insertion. This decision does not make a difference in our probabilistic model since insertion probabilities are normalized not per grammar node but per nonterminal; meanwhile, it helps us avoid the bookkeeping of distinguishing between roots of auxiliary trees and the other nodes along the spine which is more convenient in sampling. Figure 4.2 illustrates our TIG-TSG transform.

Prior work by Shindo et al. (2011) used a different CFG transform. Unfortunately, in trying to map newly formed CFG categories to real tree-spans, they inflate their CFG by redundantly forming a new label for every [auxiliary tree / foot subtree] pair. The transformation that we use factors out the insertions outside the elementary trees, thus producing a leaner representation and reducing the grammar constant.[6] Our only disadvantage is that inside-outside computation during sampling no longer scales linearly in the size of the parse tree, since we must also entertain spans of insertion (spans dominated by potential [root / foot] pairs). However in practice these spans are constrained by the parse tree (especially since all nodes along any spine must be of the same label) significantly reducing our computational burden.

[6] If N_{aux} is the number of unique subtrees in the set of auxiliary trees and N_{init} is the number of unique subtrees in the set of initial trees, the size of our representation scales as $O(N_{\text{aux}} + N_{\text{init}})$ whereas they create $O(N_{\text{aux}} \cdot N_{\text{init}})$ labels.

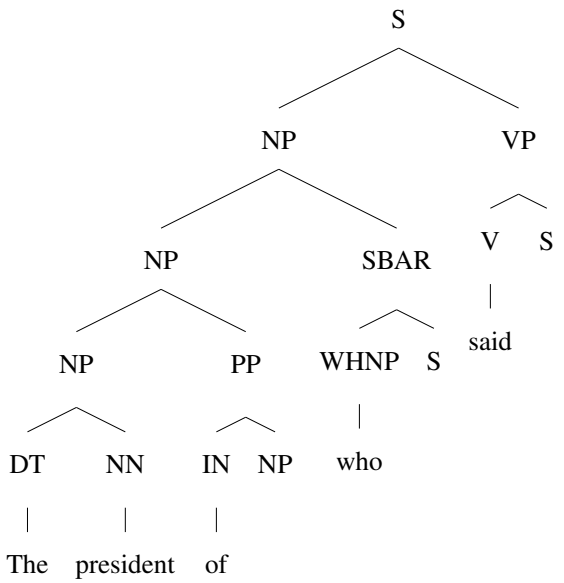


Figure 4.3: Sentence structure used for proof of concept experiment.

Proof of concept

As a proof of concept toy experiment, we list a few sentences with structures of the form in Figure 4.4, and compare grammars obtained by TSG and TIG. Given the following small corpus,

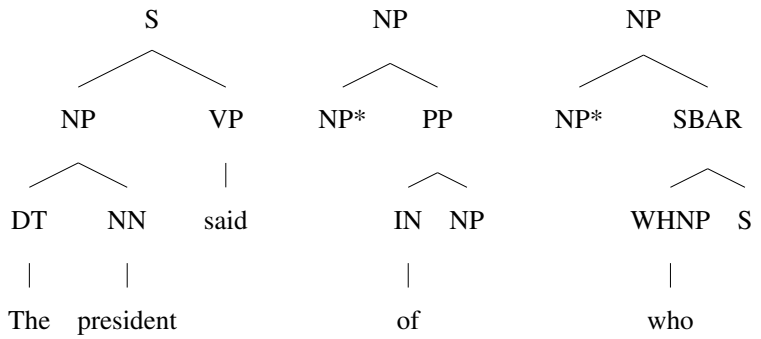
"The president said.

The president of NP said.

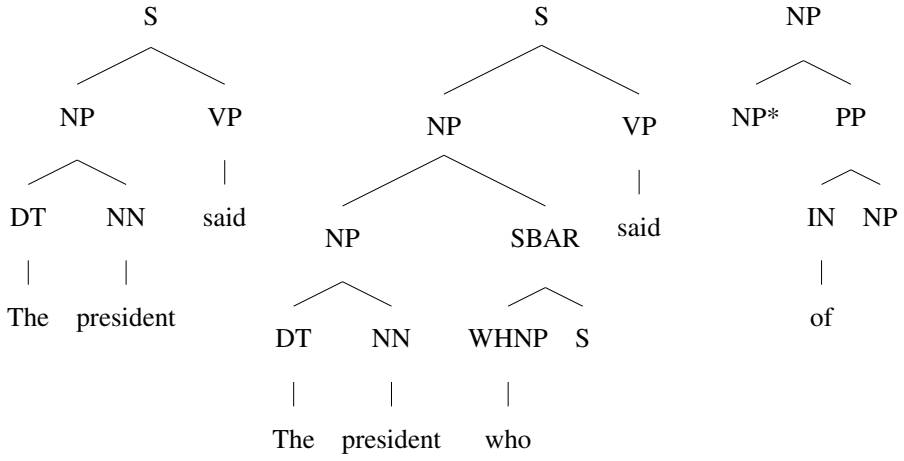
The president, who S, said.

The president of NP, who S, said."

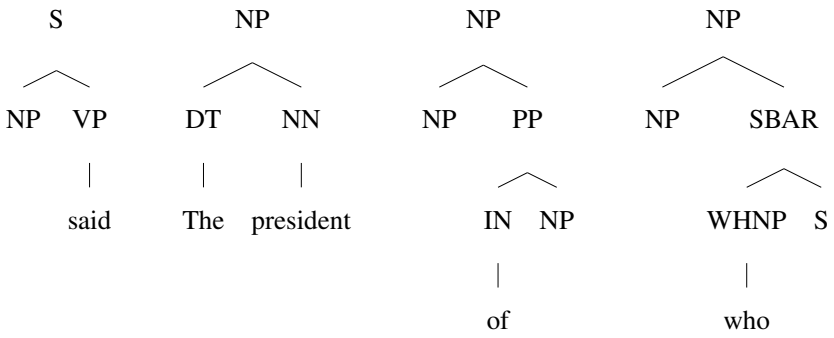
our TIG sampler converges in 1000 iterations on the desirable grammar:



If we constrain it so that it cannot use simultaneous insertion and can only support auxiliary trees of depth 1, TIG has to learn an extra specialized initial tree:



Whereas TSG, in order to model this phenomenon via substitutions, has to break the background initial tree at its NP label into two, thus introducing an unnecessary independence assumption (between "the president" and "said"), and converges on the following solution.



4.1.5 Evaluation

We use the standard Penn treebank methodology of training on sections 2–21 and testing on section 23. All our data is head-binarized and words occurring only once are mapped into unknown categories of the Berkeley parser. As has become standard, we carried out a small treebank experiment where we train on section 2, and a large one where we train on the full training set. All hyperparameters are resampled under appropriate vague Gamma and Beta priors. All reported numbers are averages over three runs. Parsing results are based on the maximum probability parse, which was obtained by sampling derivations under the transformed CFG.

Table 4.3: EVALB results after training on section 2 and testing on section 23. Note that TIG finds a compact yet rich representation. Elementary tree counts are based on ones with count greater than 1.

Model	FMeasure	# Initial Trees	# Auxiliary Trees (# Left)
TSG	77.51	6.2K	-
TIG_0	78.46	6.0K	251 (137)
TIG	78.62	5.6K	604 (334)

We compare our system (referred to as TIG) to our implementation of the TSG sys-

tem of Cohn and Blunsom (2010) (referred to as TSG) and the constrained TIG variant of Shindo et al. (2011) (referred to as TIG_0). The upshot of our experiments is that, while on the large training set all models have similar performance ($85.6, 85.3, 85.4$ for TSG, TIG_0 and TIG respectively), on the small dataset insertion helps the nonparametric model to find more compact and generalizable representations for the data, which affects parsing performance (Table 4.2). Although TIG_0 has performance close to TIG, note that TIG achieves this performance using a more succinct representation and extracting a much richer set of auxiliary trees. As a result, TIG finds many chances to apply insertions to test sentences, whereas TIG_0 depends mostly on TSG rules. If we look at the most likely derivations for the test data, TIG_0 assigns 663 insertions (351 left insertions) in the parsing of entire section 23, meanwhile TIG assigns 3924 (2100 left insertions). Examples of these linguistically sophisticated auxiliary trees that apply to test data are listed in Figure 4.5.

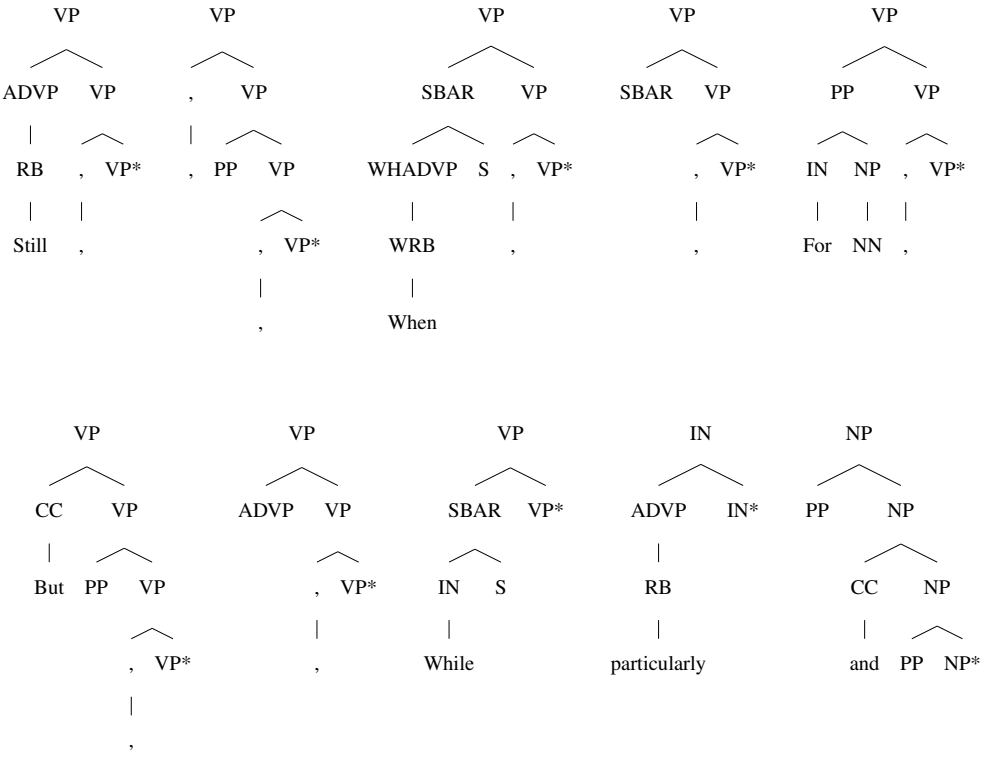


Figure 4.4: Example left auxiliary trees that occur in the top derivations for section 23. Simultaneous insertions occur most frequently for the labels VP (85 times), NNS (21 times), NNP (14 times).

Comparing the induced grammars between TIG and TSG, we find that our initial claim of gaining reuse through adjunction holds up. Figure 4.6 shows the additional counts for an elementary tree structure gained by using TIG's adjunction to strip away adverbial modifiers and punctuation.

4.1.6 Discussion

We described a Bayesian nonparametric inference scheme for extracting and estimating TIG grammars, and showed the power of TIG formalism over TSG for returning rich, generalizable, yet compact representations of data. The nonparametric inference scheme

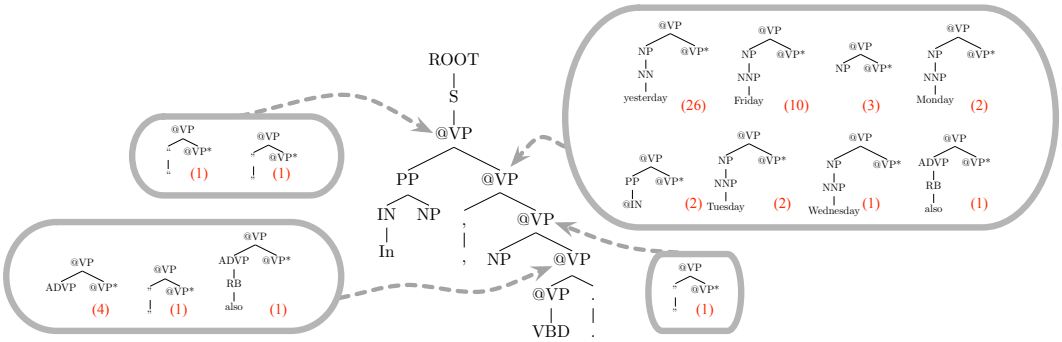


Figure 4.5: TIG finds this large elementary tree structure describing a sentence 314 times, whereas TSG can find only 242. This is due to the ability of TIG to collapse elementary tree counts by stripping away modifiers. The auxiliary trees and their counts of adjunction in the training data are shown.

presents a principled way of addressing the difficult model selection problem with TIG which has been prohibitive in this area of research. TIG still remains within context-free and both our sampling and parsing techniques are highly scalable.

Next, we will consider moving up another layer of sophistication in the hierarchy of grammars and applying the same Bayesian nonparametric grammar induction machinery to unconstrained TAG. We will tackle the complexity challenge and come up with efficient solutions for parsing.

4.2 Inference and cubic-time parsing for tree-adjoining grammars

We present a Bayesian nonparametric model for estimating a probabilistic TAG from a parsed corpus, along with approximation transformations for TAG that allow efficient pars-

ing. In line with our TIG experiments of the previous section, our TAG work shows performance improvements on the Penn Treebank and finds more compact yet linguistically rich representations of the data, but more importantly provides techniques in grammar transformation and statistical inference that make practical the use of TAG, thereby enabling further experimentation along these lines.

4.2.1 Introduction

As discussed earlier in Section 2.2, TAG's expressivity comes at the cost of greatly increased complexity. Parsing complexity for unconstrained TAG scales as $O(n^6)$, impractical as compared to CFG and TSG's $O(n^3)$. In addition, the model selection problem for TAG is significantly more complicated than for TSG since one must reason about many more combinatorial options with two types of derivation operators. This has led researchers to resort to manual or heuristic techniques; a solution has not been put forward by which a model maximizes a principled probabilistic objective.

We have seen in Section 2.3.2 that nonparametric grammar induction holds promise for solving complex model selection problems in an efficient and principled manner. Cohn and Blunsom (2010) argued that under highly expressive grammars such as TSGs where exponentially many derivations may be hypothesized of the data, the local Gibbs sampling approach is insufficient for effective inference and global blocked sampling strategies will be necessary. Due to the combinatorial nature of the grammar formalism at hand, a globally optimal solution may only be reached via a number of suboptimal local decisions, hurting mixing capabilities of inference. For TAG, this problem is only more severe due to its mild context-sensitivity and even richer combinatorial nature. Therefore in previous work,

Shindo et al. (2011) and ourselves in Section 4.1 used TIG as a kind of expressive compromise between TSG and TAG, as a substrate on which to build nonparametric inference. However TIG has the constraint of disallowing wrapping adjunction (coordination between material that falls to the left and right of the point of adjunction, such as parentheticals and quotations) as well as left adjunction along the spine of a right auxiliary tree and vice versa.

In this work we formulate a blocked sampling strategy for TAG that is effective and efficient, and prove its superiority against the local Gibbs sampling approach. We show via nonparametric inference that TAG, which contains TSG and TIG as a subset, is a better model for treebank data than both, not only in terms of improved parsing performance but also in terms of providing compact representations by using the adjunction (and particularly wrapping adjunction) operation and therefore being able to support more appropriate parametrizations. Additionally, we explain how our parameter refinement scheme for TAG allows for cubic-time parsing, which is just as efficient as TSG and TIG parsing.

4.2.2 Probabilistic model

Similar to our TIG model in Section 4.1.2, the TAG probabilistic model extends that of the TSG model of Cohn et al. (2009). However, instead of modeling independent sets of DPs for left and right auxiliary trees of every nonterminal type, we have a single set of DPs for auxiliary trees of all kinds (left, right and wrapping)

$$G_c^{\text{aux}} \sim \text{DP}(\alpha_c^{\text{aux}}, P_0^{\text{aux}}(\cdot \mid c))$$

in addition to the DPs for initial trees given in Equation 4.1. Therefore the same exchangeability properties (Equations 4.2 and 4.4) apply to our TAG model

$$p(a_j \mid \mathbf{a}_{<j}) = \frac{n_{c,a_j} + \alpha_c^{\text{aux}} P_0^{\text{aux}}(a_j \mid c)}{j - 1 + \alpha_c^{\text{aux}}} \tag{4.5}$$

We define base distributions P_0 and P_0^{aux} for initial and auxiliary trees, respectively. P_0 copies our treatment in Section 4.1.2. P_0^{aux} generates an auxiliary tree with root label c by sampling a rule from pre-estimated CFG $\tilde{P}$, flipping an unbiased coin to decide the direction of the spine (if more than a unique child was generated), making a binary decision at the spine whether to leave it as a foot node or further expand (with probability γ_c), and recurring into P_0 or P_0^{aux} appropriately for the off-spine and spinal children respectively.[7]

To illustrate, the elementary trees in Table 4.1 have probabilities defined recursively as follows

$$\begin{aligned}
P_0(\nu) &= \beta_{\text{WHNP}} \times (1 - \beta_{\text{S}}) \times \tilde{P}(\text{SBAR} \rightarrow \text{WHNP S}) \times \tilde{P}(\text{WHNP} \rightarrow \text{who}) \\
P_0^{\text{aux}}(\eta_1) &= (1 - \gamma_{\text{NP}}^{\text{aux}}) \times \beta_{\text{PP}} \times \beta_{\text{IN}} \times (1 - \beta_{\text{NP}}) \times \tilde{P}(\text{NP} \rightarrow \text{NP PP}) \times 0.5 \\
&\quad \times \tilde{P}(\text{PP} \rightarrow \text{IN NP}) \times \tilde{P}(\text{IN} \rightarrow \text{of}) \\
P_0^{\text{aux}}(\eta_2) &= \gamma_{\text{NP}}^{\text{aux}} \times \beta_{\text{SBAR}} \times P_0^{\text{aux}}(\eta_1) \times P_0(\nu) \times \tilde{P}(\text{NP} \rightarrow \text{NP SBAR}) \times 0.5
\end{aligned}$$

In any derivation for every node labeled c that is not a frontier node or the root or foot node of an auxiliary tree, we determine the number (perhaps zero) of *simultaneous adjunctions* (Schabes and Shieber, 1994) by sampling a Geometric(μ_c) variable; thus k simultaneous adjunctions would have probability $(\mu_c)^k(1 - \mu_c)$. Since we already provide

[7] This base distribution does not enforce matching root-foot labels. The main point of the base-distribution is to favor small elementary trees; therefore, we do not dwell over this issue and maintain this constraint during sampling.

simultaneous adjunction we disallow adjunction at the root of auxiliary trees. Under our current parametrization scheme (μ_c where c is merely a nonterminal label) supporting both operations would lead to spurious ambiguity due to assigning probability to distinct derivations that yield identical trees.

4.2.3 Inference

Given this model, our inference task is to explore posterior derivations underlying the data. Since TAG derivations are highly structured objects (even more so than TIG derivations), a basic sampling strategy based on local moves such as Gibbs sampling would not hold much promise. Following Cohn and Blunsom (2010), we design a blocked Metropolis-Hastings sampler that samples derivations for entire parse trees all at once in a joint fashion. We use a Goodman-transformed TAG as our proposal distribution (Goodman, 2002) that incorporates additional CFG rules to account for the possibility of backing off to the infinite base distribution P_0^{aux}, and use the parsing algorithm described by Shieber et al. (1995) for computing inside probabilities under this TAG model. This distribution represents the infinite TAG implied by the current cache of the sampler where counts from the current parse tree are removed, and only approximates the true posterior process as it disregards increments to counts caused by the reuse of elementary trees in one derivation. We account for the use of this approximation by making a Metropolis-Hastings accept/reject decision for every sampled derivation (see introduction in Section 2.1.5).

The algorithm is illustrated in Table 4.3 along with Figure 4.7. Inside probabilities are computed in a bottom-up fashion and a TAG derivation is sampled top-down (Johnson et al., 2007). The sampler visits every node of the tree in post-order ($O(n)$ operations, n

being the number of nodes), visits every node below it as a potential foot (another $O(n)$ operations), visits every mid-node in the path between the original node and the potential foot (if spine-adjunction is allowed) ($O(\log n)$ operations), and forms the appropriate chart items. The complexity is $O(n^2 \log n)$ if spine-adjunction is allowed, $O(n^2)$ otherwise.

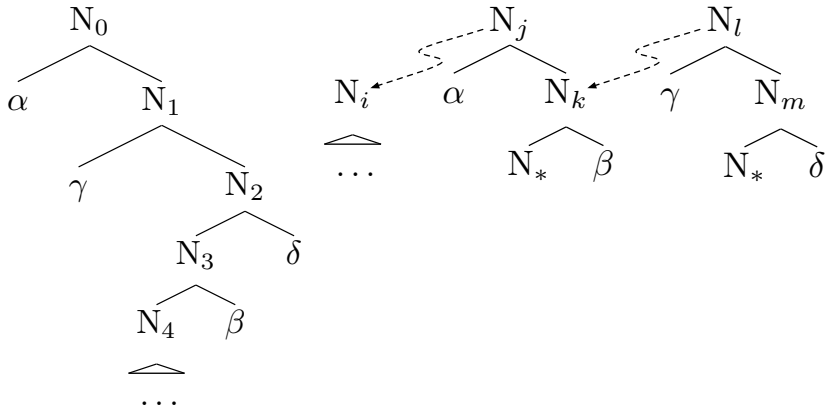


Figure 4.6: Example used for illustrating blocked sampling with TAG. On the left hand side we have a partial training tree where we highlight the particular nodes (with node labels $0,1,2,3,4$) that the sampling algorithm traverses in post-order. On the right hand side is the TAG grammar fragment that is used to parse these particular nodes: one initial tree and two *wrapping* auxiliary trees where one adjoins into the *spine* of the other for full generality of our illustration. Grammar nodes are labeled with their Goodman indices (letters i, j, k, l, m). Greek letters $\alpha, \beta, \gamma, \delta$ denote entire subtrees. We assume that a subtree in an auxiliary tree (e.g., α) parses the same subtree in a training tree.

4.2.4 Parameter refinement

During inference, adjunction probabilities are treated simplistically to facilitate convergence. Only two parameters guide adjunction: μ_c, the probability of adjunction; and $p(a_j \mid \mathbf{a}_{<j}, c)$, the probability of the particular auxiliary tree being adjoined given that there is an adjunction (see Equation 4.5). In all of this treatment, the context of an adjunction (denoted as c) is the nonterminal label such as S or NP. This is simpler than the usual treat-

Table 4.4: Computation of inside probabilities for TAG sampling. We create two types of chart items: (1) **per-node**, e.g., $\mathrm{N}_i[v]$ denoting the probability of starting at an initial subtree that has Goodman index i and generating the subtree rooted at node v, and (2) **per-path**, e.g., $\mathrm{N}_j[v\text{-}\eta]$ denoting the probability of starting at an auxiliary subtree that has Goodman index j and generating the subtree rooted at v minus the subtree rooted at η. Above, c denotes the context of adjunction, which is the nonterminal label of the node of adjunction (here, N), μ_c is the probability of adjunction, $n_{c,a}$ is the count of the auxiliary tree a, and $n_c = \sum_a n_{c,a}$ is total number of adjunctions at context c. The function $\pi(\cdot)$ retrieves the inside probability corresponding to an item.

Chart item	Why made?	Inside probability
N_i[4]	By assumption.	–
N_k[3-4]	N_*[4] and β	$(1-\mu_c)\times\pi(\beta)$
N_m[2-3]	N_*[3] and δ	$(1-\mu_c)\times\pi(\delta)$
N_l[1-3]	γ and N_m[2-3]	$(1-\mu_c)\times\pi(\gamma)\times\pi(\mathrm{N}_m[2\text{-}3])$
$\mathrm{N}_{\mathrm{aux}}$[1-3]	N_l[1-3]	$n_{c,a_l}/(n_c+\alpha_c^{aux})\times\pi(\mathrm{N}_l[1\text{-}3])$
N_k[1-4]	$\mathrm{N}_{\mathrm{aux}}$[1-3] and N_k[3-4]	$\mu_c\times\pi(\mathrm{N}_{\mathrm{aux}}[1\text{-}3])\times\pi(\mathrm{N}_k[3\text{-}4])$
N_j[0-4]	α and N_k[1-4]	$(1-\mu_c)\times\pi(\alpha)\times\pi(\mathrm{N}_k[1\text{-}4])$
$\mathrm{N}_{\mathrm{aux}}$[0-4]	N_j[0-4]	$n_{c,a_j}/(n_c+\alpha_c^{aux})\times\pi(\mathrm{N}_j[0\text{-}4])$
N_i[0]	$\mathrm{N}_{\mathrm{aux}}$[0-4] and N_i[4]	$\mu_c\times\pi(\mathrm{N}_{\mathrm{aux}}[0\text{-}4])\times\pi(\mathrm{N}_i[4])$

ment of adjunction probability in TAG literature where c is a unique identifier for the node at which the adjunction occurs as shown in Equation 2.11. We choose the former treatment for simplicity since using the latter form necessitates using hierarchical DP models as opposed to simple and flat DP models under which convergence is typically achieved more easily. However it is possible to experiment with further refinement schemes at parsing time. Once the sampler converges on a grammar, we can re-estimate its adjunction probabilities. Using the well-known $O(n^6)$ TAG parsing algorithm (Shieber et al., 1995)

we experimented with various refinement schemes — ranging from full node identifiers, to Goodman index identifiers of the subtree below the adjunction (Hwa, 1998), to simple nonterminal labels — and found that using Goodman index identifiers as c is the best performing option. Interestingly, this particular refinement scheme also allows for fast cubic-time parsing, which we achieve by approximating the TAG by a TSG with little loss of coverage (*no* loss of coverage under special conditions that we find are often satisfied) and negligible increase in grammar size, as discussed in the next section.

4.2.5 Cubic-time parsing

MCMC training results in a list of sufficient statistics of the final derivation that the TAG sampler converges upon after a number of iterations. Basically, these are the list of initial and auxiliary trees, their cumulative counts over the training data, and their adjunction statistics. An adjunction statistic is listed as follows. If α is any elementary tree, and β is an auxiliary tree that adjoins n times at node v of α that is uniquely reachable at path p, we write "$\alpha \xleftarrow{p} \beta$ (n times)". We denote v alternatively as $\alpha[p]$.

Now imagine that we end up with a small grammar that consists of one initial tree α and two auxiliary trees β and γ, and the following adjunctions occurring between them

$$\alpha \xleftarrow{p} \beta \ (n \text{ times})$$

$$\alpha \xleftarrow{p} \gamma \ (m \text{ times})$$

$$\beta \xleftarrow{q} \gamma \ (k \text{ times})$$

as shown in Figure 4.8. Assume that α itself occurs $l > n+m$ times in total so that there is nonzero probability of no adjunction anywhere within α. Also assume that the node uniquely identified by $\alpha[p]$ has Goodman index i, which we denote as $i = G(\alpha[p])$.

The general idea of this TAG-TSG approximation is that, for any auxiliary tree that adjoins at a node v with Goodman index i, we create an initial tree out of it where the root and foot nodes of the auxiliary tree are both replaced by i. Further, we split the subtree rooted at v from its parent and rename the substitution site that is newly created at v as i as well. (See Figure 4.8.) We can separate the foot subtree from the rest of the initial tree since it is completely remembered by any adjoined auxiliary trees due to the nature of our refinement scheme. However this method fails for adjunctions that occur at *spinal* nodes of auxiliary trees that have foot nodes below them since we would not know in which order to do the initial tree creation. However when the *spine-adjunction relation* is amenable to a *topological sort* (as is the case in Figure 4.8), we can apply the method by going in this order and doing some extra bookkeeping: updating the list of Goodman indices and re-directing adjunctions as we go along. When there is no such topological sort, we can approximate the TAG by heuristically dropping low-frequency adjunctions that introduce cycles.[8]

The algorithm is illustrated in Figure 4.8. In (1) we see the original TAG grammar and its adjunctions (n,m,k are adjunction counts). Note that the adjunction relation has a topological sort of α,β,γ. We process auxiliary trees in this order and iteratively remove their adjunctions by creating specialized initial tree duplicates. In (2) we first visit β, which has adjunctions into α at the node denoted $\alpha[p]$ where p is the unique path from the root to this node. We retrieve the Goodman index of this node $i = G(\alpha[p])$, split the

[8]We found that, on average, about half of our grammars have a topological sort of their spine-adjunctions. (On average fewer than 100 spine adjunctions even exist.) When no such sort exists, only a few low-frequency adjunctions have to be removed to eliminate cycles.

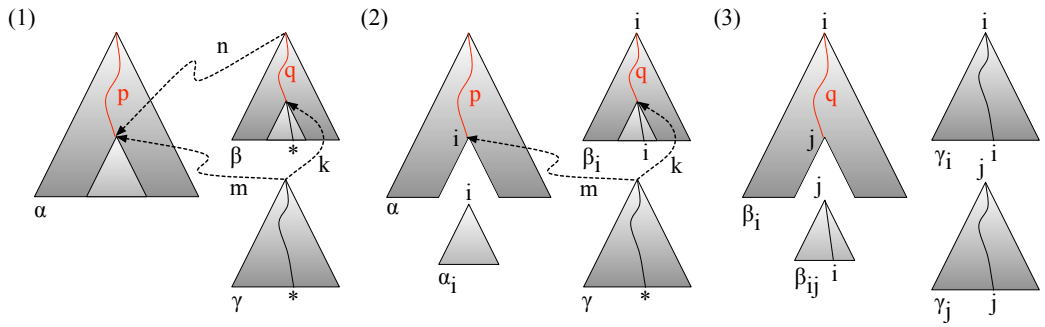


Figure 4.7: TAG to TSG transformation algorithm. By removing adjunctions in the correct order we end up with a larger yet adjunction-free TSG.

subtree rooted at this node as a new initial tree α_i, relabel its root as i, and rename the newly-created substitution site at $\alpha[p]$ as i. Since β has only this adjunction, we replace it with initial tree version β_i where root/foot labels of β are replaced with i (alternatively create new branches that extend from the root/foot), and update all adjunctions into β as being into β_i.[9] In (3) we visit γ which now has adjunctions into α and β_i. For the $\alpha[p]$ adjunction we create γ_i the same way we created β_i but this time we cannot remove γ as it still has an adjunction into β_i. We retrieve the Goodman index of the node of adjunction $j = G(\beta_i[q])$, split the subtree rooted at this node as new initial tree β_{ij}, relabel its root as j, and rename the newly-created substitution site at $\beta_i[q]$ as j. Since γ now has only this adjunction left, we remove it by also creating initial tree version γ_j where root/foot labels of γ are replaced with j. At this point we have an adjunction-free TSG with elementary trees (and counts) $\alpha(l), \alpha_i(l), \beta_i(n), \beta_{ij}(n), \gamma_i(m), \gamma_j(k)$ where l is the count of initial tree α. These counts, when they are normalized, lead to the appropriate adjunction probability

[9] If β adjoined into multiple initial trees, multiple copies of β would be necessary, one for each Goodman index of the adjunction site in an elementary tree into which β adjoins. The duplication is exemplified in the discussion of γ next.

refinement scheme of $\mu_c \times p(a_j \mid \mathbf{a}_{<j}, c)$ where c is the Goodman index.

Although this algorithm increases grammar size, the sparsity of the nonparametric solution ensures that the increase is almost negligible: on average the final Goodman-transformed CFG has 173.9K rules for TSG, 189.2K for TAG. Figure 4.9 demonstrates the comparable Viterbi parsing times for TSG and TAG.

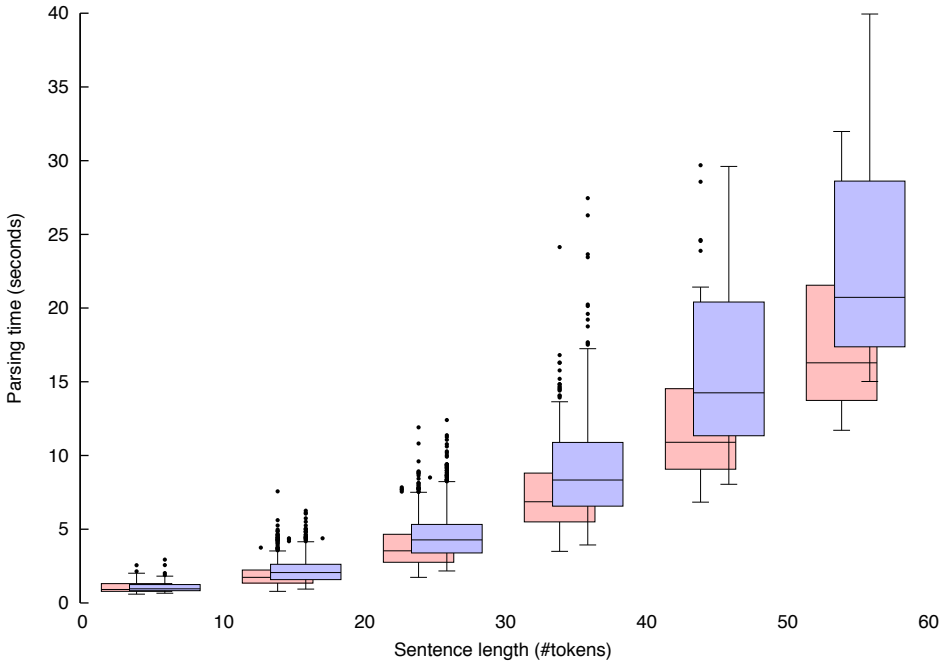


Figure 4.8: Nonparametric TAG (blue) parsing is efficient and incurs only a small increase in parsing time compared to nonparametric TSG (red).

4.2.6 Evaluation

We use the standard Penn treebank methodology of training on sections 2–21 and testing on section 23. All our data is head-binarized, and words occurring only once are mapped into unknown categories of the Berkeley parser. All hyperparameters are resampled under appropriate vague Gamma and Beta priors. Samplers are run 1000 iterations

each; all reported numbers are averages over five runs. For simplicity, parsing results are based on the maximum probability derivation (Viterbi algorithm).

In Table 4.4, we compare TAG inference schemes and TSG. $\text{TAG}_{\text{Gibbs}}$ operates by locally adding/removing potential adjunctions, similar to the TSG sampler of Cohn et al. (2009). TAG' is the $O(n^2)$ algorithm that disallows spine adjunction. We see that TAG' has the best parsing performance, while TAG provides the most compact representation.

Table 4.5: EVALB results. Note that the Gibbs sampler for TAG has poor performance and provides no grammar compaction due to its lack of convergence.

model	*F measure*	*# initial trees*	*# auxiliary trees*
TSG	84.15	69.5K	-
$\text{TAG}_{\text{Gibbs}}$	82.47	69.9K	1.7K
TAG'	**84.87**	66.4K	1.5K
TAG	84.82	**66.4K**	**1.4K**

4.2.7 Discussion

We described a Bayesian nonparametric inference scheme for estimating TAG grammars and showed the power of TAG formalism over TSG for returning rich, generalizable, yet compact representations of data. The nonparametric inference scheme presents a principled way of addressing the difficult model selection problem with TAG, which has been prohibitive in this area of research. Our sampler has near quadratic-time efficiency, and our heuristic parsing approach remains context-free allowing for fast cubic-time parsing, so that our overall parsing framework is highly scalable; therefore, it would be feasible

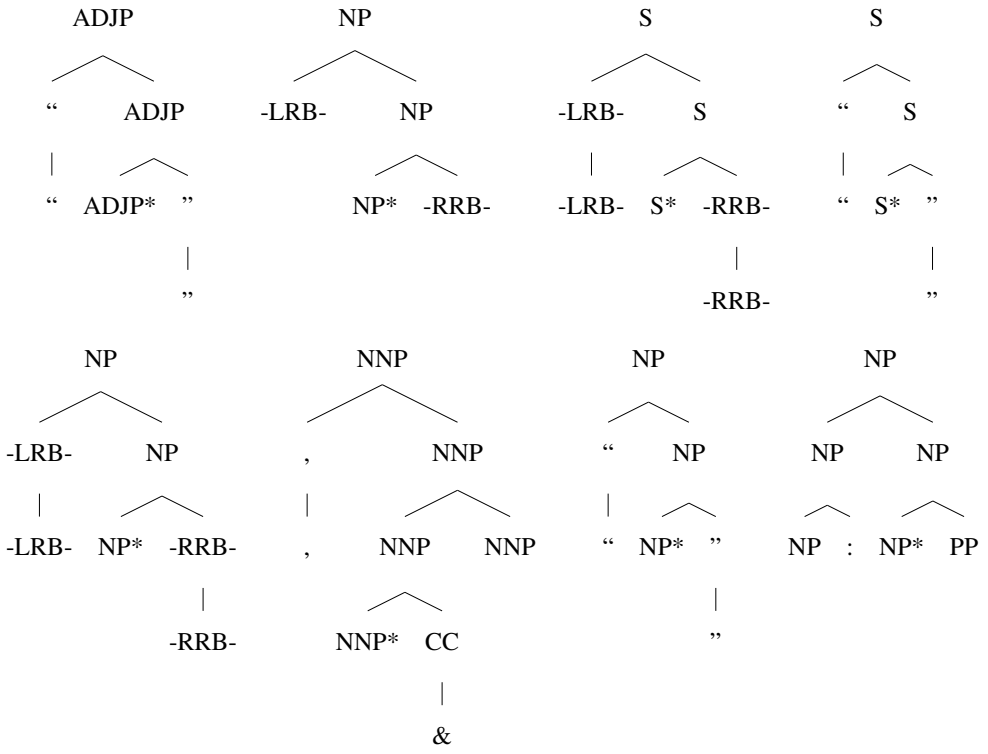


Figure 4.9: Example wrapping trees from estimated TAGs.

to perform grammar induction on a larger scale. In the next chapter, we address the very question of using these induction techniques to discover complex grammatical structure in web-data.

Table 4.6: Grammar analysis for an estimated TAG, categorized by label. Only those having more than 100 adjunctions are shown, binarization variables are denoted with overline. A total number of 98 wrapping adjunctions (9 unique wrapping trees) and 118 spine adjunctions occur.

label	*#adjunctions (spine)*	*average depth*	*#lexicalized trees*	*#left trees*	*#right trees*	*#wrapping trees*
$\overline{\text{VP}}$	4532 (23)	1.06	45	22	65	0
$\overline{\text{NP}}$	2891 (46)	1.71	68	94	13	1
$\overline{\text{NN}}$	2160 (3)	1.08	85	16	110	0
$\overline{\text{NNP}}$	1478 (2)	1.12	90	19	90	0
$\overline{\text{NNS}}$	1217 (1)	1.10	43	9	60	0
$\overline{\text{VBN}}$	1121 (1)	1.05	6	18	0	0
$\overline{\text{VBD}}$	976 (0)	1.0	16	25	0	0
NP	937 (0)	3.0	1	5	0	0
$\overline{\text{VB}}$	870 (0)	1.02	14	31	4	0
$\overline{\text{S}}$	823 (11)	1.48	42	36	35	3
$\overline{\text{VP}}$	803 (17)	1.42	29	37	10	0
$\overline{\text{VBZ}}$	764 (0)	1.10	16	28	0	0
$\overline{\text{VBP}}$	685 (0)	1.0	13	22	0	0
$\overline{\text{VBG}}$	554 (0)	1.0	5	12	0	0
$\overline{\text{IN}}$	315 (0)	1.0	9	12	5	0
VP	307 (0)	2.0	1	0	2	0
$\overline{\text{MD}}$	306 (0)	1.0	9	17	1	0
$\overline{\text{POS}}$	239 (2)	1.23	30	0	42	0
$\overline{\text{JJ}}$	210 (0)	1.1	14	11	9	0
$\overline{\text{IN}}$	169 (0)	1.08	10	9	3	0
$\overline{\text{IN}}$	127 (0)	1.0	24	24	7	0
$\overline{\text{VP}}$	126 (0)	1.19	19	16	20	0
$\overline{\text{CD}}$	108 (0)	1.15	10	3	10	0
$\overline{\text{RB}}$	102 (0)	1.0	9	8	8	0
total	23320 (118)	1.25	824	743	683	9

Chapter 5

Nonparametric Solutions and Web Data

We have seen how the nonparametric Bayesian grammar induction framework can provide a solution to the complex joint problem of model selection and estimation for grammar formalisms with rich linguistic structure, namely TIG and TAG. In this chapter we will look at the effectiveness of this framework by testing it in the real application domain of extractive sentence compression and in web-scale data settings.

Firstly, we extend the nonparametric Bayesian TSG model of Cohn et al. (2009) to the synchronous setting (similar to Section 3.1) where each elementary tree *pair* is composed of a source and a target elementary tree, forming a synchronous TSG (STSG).[1] Using the familiar rich-gets-richer dynamics of Dirichlet Processes, one can extract many interpretable compression patterns that have high frequencies and therefore generalize well to testing examples, once again solving the model selection and estimation problems all at once. We

[1] We choose to extend TSG instead of TIG (or TAG) for the sake of simplicity, as TSGs can also capture a variety of linguistic phenomena themselves. However our arguments have straightforward parallels for the higher grammar formalisms of TIG and TAG as well.

demonstrate this by comparing our Bayesian STSG to heuristically extracted and parametrically trained STSG baselines on a small hand-curated sentence compression corpus. Our experiments not only prove the superior performance of the Bayesian approach, but also highlight the benefits of solving the model selection problem via principled, non-heuristic means.

Secondly, we test the Bayesian STSG model on the web-scale sentence compression corpus that we obtained from Wikipedia in Section 3.1 in order to support the claim that the model selection benefits will truly shine in larger data set domains. Since the model selection process rests on re-use of patterns, larger data sets will allow us to induce compact yet informed grammars that provide good performance and rich linguistic analyses. Our experiments show that, similar to our result in Section 3.1, our Bayesian model trained on Wikipedia compressions has state-of-the-art performance even in out-of-domain settings. However, the benefit of the current model is that its linguistic sophistication is adaptive and determined via data-driven model selection as opposed to blind lexicalization. This provides not only interpretable but also compact grammar representations.[2]

5.1 Bayesian synchronous tree-substitution grammar induction and its application to sentence compression

Given an aligned corpus of tree pairs, we might want to learn a mapping between the paired trees. Such induction of tree mappings has application in a variety of NLP tasks that

[2]This chapter is substantially based on previously published papers, with text used by permission of the authors (Yamangil and Shieber, 2010).

are based on parallel text, including machine translation, paraphrase, and sentence compression. The induced tree mappings can be expressed by synchronous grammars. Where the tree pairs are isomorphic, synchronous context-free grammars (SCFG) may suffice, but in general, non-isomorphism can make the problem of rule extraction difficult (Galley and McKeown, 2007). More expressive formalisms such as synchronous tree-substitution (Eisner, 2003) or tree-adjoining grammars may better capture the pairings.

In this work, we explore techniques for inducing synchronous tree-substitution grammars (STSG) using as a test-bed application sentence compression – the task of summarizing a sentence while retaining most of the informational content and remaining grammatical (Jing, 2000). An example sentence pair is the following:

- Like FaceLift, much of ATM's screen performance depends on the underlying application.
- ATM's screen performance depends on the underlying application.

where the underlined words were deleted.[3]

Learning an STSG from aligned trees is tantamount to determining a segmentation of the trees into elementary trees of the grammar along with an alignment of the elementary trees (see Figure 5.1 for an example of such a segmentation), followed by estimation of the weights for the extracted tree pairs.[4] These elementary tree pairs serve as the *rules* of the extracted grammar. For SCFG, segmentation is trivial — each parent with its immediate

[3]Once again, we will narrow in on *extractive* sentence compression which permits word deletions only (no re-ordering or replacements).

[4]Throughout the chapter we will use the word STSG to refer to the tree-to-tree version of the formalism, although the string-to-tree version is also commonly used.

children is an elementary tree — but the formalism then restricts us to deriving isomorphic tree pairs. STSG is much more expressive, especially if we allow some elementary trees on the source or target side to be unsynchronized, so that insertions and deletions can be modeled, but the segmentation and alignment problems become nontrivial.

Previous approaches to this problem have treated the two steps — grammar extraction and weight estimation — with a variety of methods. One approach is to use word alignments (where these can be reliably estimated, as in our testbed application) to align subtrees and extract rules (Och and Ney, 2004; Galley et al., 2004) but this leaves open the question of finding the right level of generality of the rules — how deep the rules should be and how much lexicalization they should involve — necessitating resorting to heuristics such as *minimality* of rules, and leading to large grammars. Once a given set of rules is extracted, weights can be imputed using a discriminative approach to maximize the (joint or conditional) likelihood or the classification margin in the training data (taking or not taking into account the derivational ambiguity). This option leverages a large amount of manual domain knowledge engineering and is not in general amenable to latent variable problems.

A simpler alternative to this two step approach is to use a generative model of synchronous derivation and simultaneously segment and weight the elementary tree pairs to maximize the probability of the training data under that model; the simplest exemplar of this approach uses expectation maximization (EM) (Dempster et al., 1977) (see Section 2.7 for an introduction). This approach has two frailties. First, EM search over the space of all possible rules is computationally impractical. Second, even if such a search were practical, the method is degenerate, pushing the probability mass towards larger rules in order to better approximate the empirical distribution of the data (Goldwater et al., 2006; DeNero

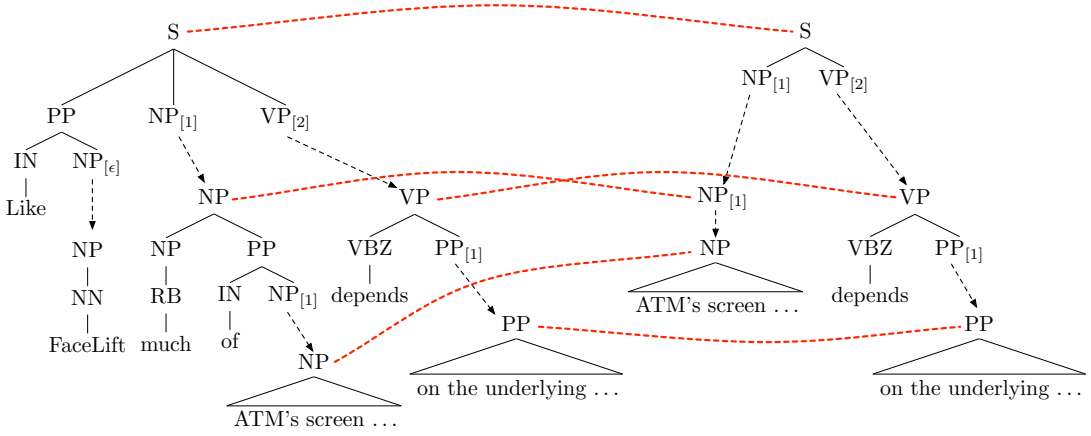


Figure 5.1: A portion of an STSG derivation of the example sentence and its extractive compression.

et al., 2006). Indeed, the optimal grammar would be one in which each tree pair in the training data is its own rule. Therefore, proposals for using EM for this task start with a precomputed subset of rules, and with EM used just to assign weights within this grammar. In summary, previous methods suffer from problems of *narrowness* of search, having to restrict the space of possible rules, and *overfitting* in preferring overly specific grammars.

We pursue the use of hierarchical probabilistic models incorporating sparse priors to simultaneously solve both the narrowness and overfitting problems. Such models have been used as generative solutions to several other segmentation problems, ranging from word segmentation (Goldwater et al., 2006), to parsing (Cohn et al., 2009; Post and Gildea, 2009) and machine translation (DeNero et al., 2008; Cohn and Blunsom, 2009; Liu and Gildea, 2009). Segmentation is achieved by introducing a prior bias towards grammars that are compact representations of the data, namely by enforcing *simplicity* and *sparsity*: preferring simple rules (smaller segments) unless the use of a complex rule is evidenced

by the data (through repetition), and thus mitigating the overfitting problem. As we previously described in Section 2.3.2, a Dirichlet process (DP) prior is typically used to achieve this interplay. Interestingly, sampling-based nonparametric inference further allows the possibility of searching over the infinite space of grammars (and, in machine translation, possible word alignments), thus side-stepping the narrowness problem outlined above as well.

In this work, we use an extension of the aforementioned models of generative segmentation for STSG induction, and describe an MCMC algorithm (see Section 2.1.5 for an introduction) for posterior inference under this model that is tailored to the task of extractive sentence compression. This task is characterized by the availability of word alignments, providing a clean testbed for investigating the effects of grammar extraction. We achieve substantial improvements against a number of baselines including EM, support vector machine (SVM) based discriminative training, and variational Bayes (VB) (see Section 2.1.6 for an introduction to variational inference techniques). By comparing our method to a range of other methods that are subject differentially to the two problems, we can show that both play an important role in performance limitations, and that our method helps address both as well. Our results are thus not only encouraging for grammar estimation using sparse priors but also illustrative of the merits of nonparametric inference over the space of grammars as opposed to sparse parametric inference with a fixed grammar.

5.1.1 The STSG model

Synchronous tree-substitution grammar is a formalism for synchronously generating a pair of non-isomorphic source and target trees (Eisner, 2003). Every grammar rule is a pair

of elementary trees aligned at the leaf level at their frontier nodes, which we will denote using the form

$$c_s/c_t \rightarrow e_s/e_t, \quad \gamma$$

(indices s for source, t for target) where c_s, c_t are root nonterminals of the elementary trees e_s, e_t respectively and γ is a 1-to-1 correspondence between the frontier nodes in e_s and e_t. For example, the rule

$$\begin{aligned} \text{S}/\text{S} \quad \rightarrow \quad & (\text{S (PP (IN Like) NP}_{[\varepsilon]}) \text{ NP}_{[1]} \text{ VP}_{[2]})/ \\ & (\text{S NP}_{[1]} \text{ VP}_{[2]}) \end{aligned}$$

can be used to delete a subtree rooted at PP (see Figure 5.1). We use square bracketed indices to represent the alignment γ of frontier nodes — $\text{NP}_{[1]}$ aligns with $\text{NP}_{[1]}$, $\text{VP}_{[2]}$ aligns with $\text{VP}_{[2]}$, $\text{NP}_{[\varepsilon]}$ aligns with the special symbol ε denoting a deletion from the source tree. Symmetrically ε-aligned target nodes are used to represent insertions into the target tree. Similarly, the rule

$$\text{NP}/\varepsilon \quad \rightarrow \quad (\text{NP (NN FaceLift)})/\varepsilon$$

can be used to continue deriving the deleted subtree. See Figure 5.1 for an example of how an STSG with these rules would operate in synchronously generating our example sentence pair.

STSG is a convenient choice of formalism for a number of reasons. First, it eliminates the isomorphism and strong independence assumptions of SCFGs. Second, the ability to have rules deeper than one level provides a principled way of modeling lexicalization, whose importance has been emphasized (Galley and McKeown, 2007; Yamangil and

Nelken, 2008). Third, we may have our STSG operate on trees instead of sentences, which allows for efficient parsing algorithms, as well as providing syntactic analyses for our predictions, which is desirable for automatic evaluation purposes.

A straightforward extension of the popular EM algorithm for probabilistic context free grammars (PCFG), the inside-outside algorithm (Lari and Young, 1990), can be used to estimate the rule weights of a given unweighted STSG based on a corpus of parallel parse trees $\mathbf{t} = t_1, \ldots, t_N$ where $t_n = t_{n,s}/t_{n,t}$ for $n = 1, \ldots, N$. Similarly, an extension of the Viterbi algorithm is available for finding the maximum probability derivation, useful for predicting the target analysis $t_{N+1,t}$ for a test instance $t_{N+1,s}$. (Eisner, 2003) However, as noted earlier, EM is subject to the narrowness and overfitting problems.

The Bayesian generative process

Both of these issues can be addressed by taking a Bayesian nonparametric approach, namely, assuming that the elementary tree pairs are sampled from an independent collection of Dirichlet process (DP) priors. We design the following generative process that is similar to the treatment of Bayesian nonparametric TSG by Cohn et al. (2009) that we previously presented in Section 2.3.2. For all pairs of root labels $c = c_s/c_t$ that we consider, where up to one of c_s or c_t can be ε (e.g., S / S, NP / ε), we sample a sparse discrete distribution G_c over infinitely many elementary tree pairs $e = e_s/e_t$ sharing the common root c from a DP

$$G_c \quad \sim \quad \mathrm{DP}(\alpha_c, P_0(\cdot \mid c)) \tag{5.1}$$

where the DP has the concentration parameter α_c controlling the sparsity of G_c, and the base distribution $P_0(\cdot \mid c)$ is a distribution over novel elementary tree pairs that we describe more fully shortly.

We then sample a sequence of elementary tree pairs to serve as a derivation for each observed derived tree pair. For each $n = 1, \ldots, N$, we sample elementary tree pairs $e_n = e_{n,1}, \ldots, e_{n,d_n}$ in a derivation sequence (where d_n is the number of rules used in the derivation), consulting G_c whenever an elementary tree pair with root c is to be sampled.

$$e \overset{\text{iid}}{\sim} G_c, \quad \text{for all } e \text{ whose root label is } c$$

Given the derivation sequence e_n, a tree pair t_n is determined, that is,

$$p(t_n \mid e_n) = \begin{cases} 1 & e_{n,1}, \ldots, e_{n,d_n} \text{ derives } t_n \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

The hyperparameters α_c can be incorporated into the generative model as random variables; however, we opt to fix these at various constants to investigate different levels of sparsity.

For the base distribution $P_0(\cdot \mid c)$ there are a variety of choices; we used the following simple scenario. (We take $c = c_s/c_t$.)

Synchronous rules For the case where neither c_s nor c_t are the special symbol ε, the base distribution first generates e_s and e_t independently, and then samples an alignment between the frontier nodes. Given a nonterminal, an elementary tree is generated by first making a decision to expand the nonterminal (with probability β_c) or to leave it as a frontier node ($1 - \beta_c$). If the decision to expand was made, we sample an appropriate rule from a PCFG which we estimate ahead of time from the training corpus. We expand the nonterminal using this rule, and then repeat the same procedure for every child generated that is a nonterminal until there are no generated nonterminal children left. This is done independently for both e_s and e_t. Finally,

we sample an alignment between the frontier nodes uniformly at random out of all possible alignments.

Deletion/insertion rules If $c_t = \varepsilon$, that is, we have a deletion rule, we need to generate $e = e_s/\varepsilon$. (The insertion rule case is symmetric.) The base distribution generates e_s using the same process described for synchronous rules above. Then with probability 1 we align all frontier nodes in e_s with ε. In essence, this process generates TSG rules, rather than STSG rules, which are used to cover deleted (or inserted) subtrees.

This simple base distribution does nothing to enforce an alignment between the internal nodes of e_s and e_t. One may come up with more sophisticated base distributions. However the main point of the base distribution is to encode a controllable preference towards simpler rules; we therefore make the simplest possible assumption.

Posterior inference via Gibbs sampling

Assuming fixed hyperparameters $\alpha = \{\alpha_c\}$ and $\beta = \{\beta_c\}$, our inference problem is to find the posterior distribution of the derivation sequences $\mathbf{e} = e_1, \ldots, e_N$ given the observations $\mathbf{t} = t_1, \ldots, t_N$. Applying Bayes' rule, we have

$$p(\mathbf{e} \mid \mathbf{t}) \quad \propto \quad p(\mathbf{t} \mid \mathbf{e}) p(\mathbf{e}) \tag{5.3}$$

where $p(\mathbf{t} \mid \mathbf{e})$ is an indicator function $\mathbb{1}[\mathbf{e} \text{ derives } \mathbf{t}]$ (5.2) which does not depend on G_c, and $p(\mathbf{e})$ can be obtained by collapsing G_c for all c.

Consider repeatedly generating elementary tree pairs $e_1, \ldots, e_i$, all with the same root c, iid from G_c. Integrating over G_c, the e_i become dependent. The conditional prior of the

i-th elementary tree pair given previously generated ones $e_{<i} = e_1, \ldots, e_{i-1}$ is given by

$$p(e_i \mid e_{<i}) \quad = \quad \frac{n_{e_i} + \alpha_c P_0(e_i \mid c)}{i - 1 + \alpha_c} \tag{5.4}$$

where n_{e_i} denotes the number of times e_i occurs in $e_{<i}$. Since the collapsed model is exchangeable in the e_i, this formula forms the backbone of the inference procedure that we describe next. It also makes clear DP's inductive bias to reuse elementary tree pairs.

We use Gibbs sampling (Geman and Geman, 1984), a Markov chain Monte Carlo (MCMC) method, to sample from the posterior (Equation 5.3). A derivation $\mathbf{e}$ of the corpus $\mathbf{t}$ is completely specified by an alignment between source nodes and the corresponding target nodes (as well as ε on either side), which we take to be the state of the sampler. We start at a random derivation of the corpus, and at every iteration resample a derivation by amending the current one through local changes made at the node level, in the style of Goldwater et al. (2006).

Our sampling updates are extensions of those used by Cohn and Blunsom (2009) in MT, but are tailored to our task of extractive sentence compression. In our task, no target node can align with ε (which would indicate a subtree insertion), and barring unary branches no source node i can align with two different target nodes j and j' at the same time (indicating a tree expansion). Rather, the configurations of interest are those in which only source nodes i can align with ε, and two source nodes i and i' can align with the same target node j. Thus, the alignments of interest are not arbitrary relations, but (partial) functions from nodes in e_s to nodes in e_t or ε. We therefore sample in the direction from source to target. In particular, we visit every tree pair and each of its source nodes i, and update its alignment by selecting between and within two choices: (a) unaligned, (b) aligned with some target node j or ε. The number of possibilities j in (b) is significantly limited, firstly by the word

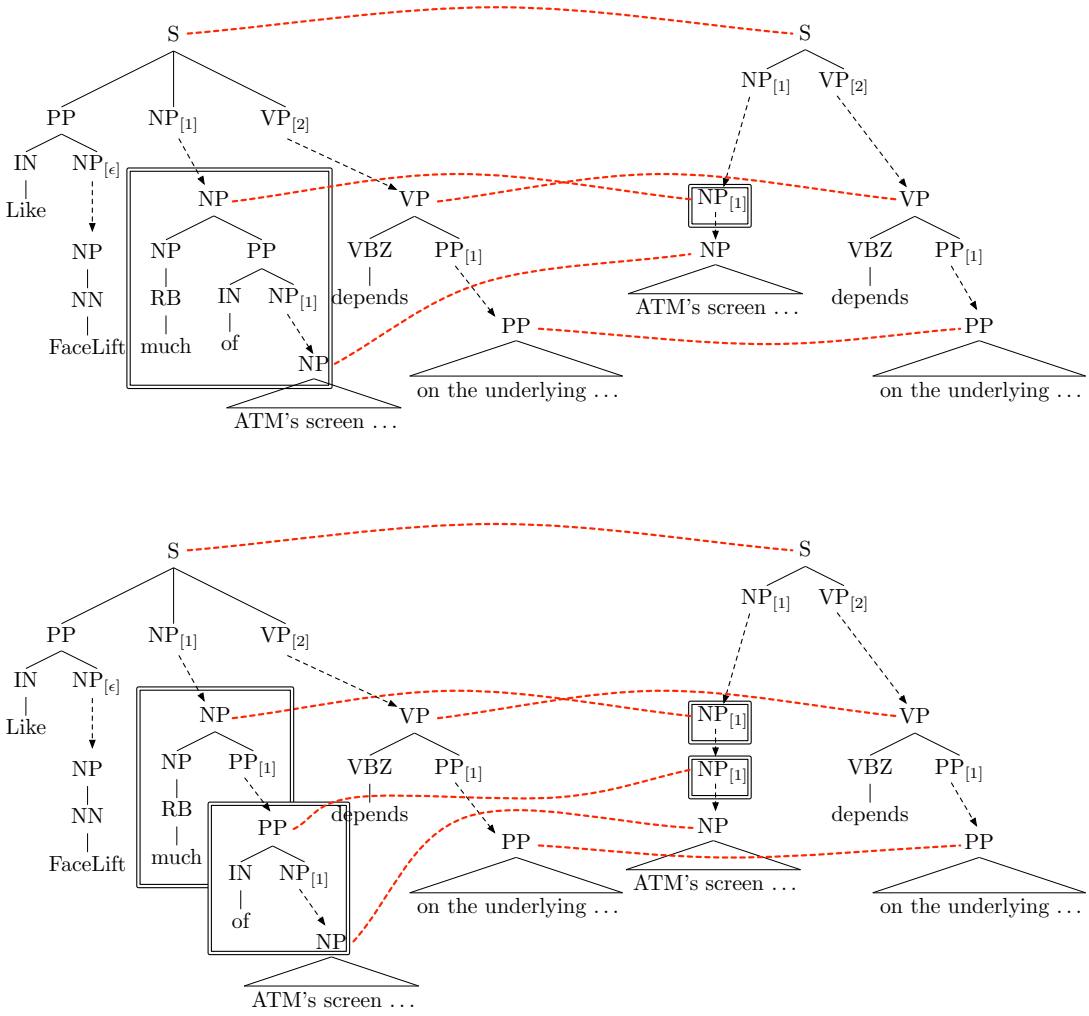


Figure 5.2: Gibbs sampling updates. We illustrate a sampler move to align/unalign a source node with a target node.

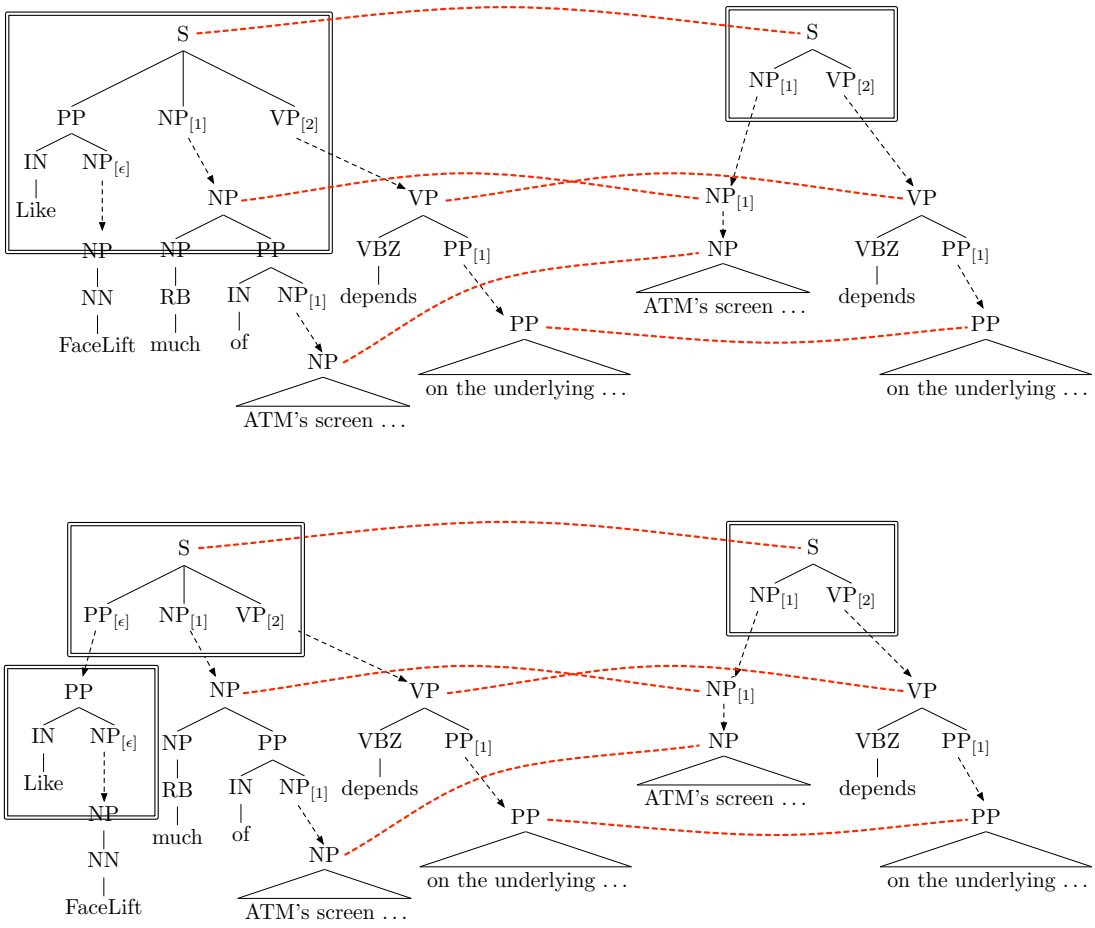


Figure 5.3: Gibbs sampling updates. We illustrate a sampler move to split/merge a deletion rule via aligning with ϵ.

alignment (for instance, a source node dominating a deleted subspan cannot be aligned with a target node), and secondly by the current alignment of other nearby aligned source nodes. See Cohn and Blunsom (2009) for details of matching spans under tree constraints.

More formally, let e_M be the elementary tree pair rooted at the closest aligned ancestor i' of node i when it is unaligned; and let e_A and e_B be the elementary tree pairs rooted at i' and i respectively when i is aligned with some target node j or ε. Then, by exchangeability of the elementary trees sharing the same root label, and using Equation 5.4, we have

$$p(\text{unalign}) = \frac{n_{e_M} + \alpha_{c_M} P_0(e_M \mid c_M)}{n_{c_M} + \alpha_{c_M}} \tag{5.5}$$

$$p(\text{align with } j) = \frac{n_{e_A} + \alpha_{c_A} P_0(e_A \mid c_A)}{n_{c_A} + \alpha_{c_A}} \times \frac{n_{e_B} + \alpha_{c_B} P_0(e_B \mid c_B)}{n_{c_B} + \alpha_{c_B}} \tag{5.6}$$

where the counts $n_{e_\cdot}, n_{c_\cdot}$ are with respect to the current derivation of the rest of the corpus; except for n_{e_B}, n_{c_B} we also make sure to account for having generated e_A. See Figure 5.2 for an illustration of the sampling updates.

It is important to note that the sampler described can move from any derivation to any other derivation with positive probability (if only, for example, by virtue of fully merging and then resegmenting), which guarantees convergence to the posterior (Equation 5.3). However some of these transition probabilities can be extremely small due to passing through low probability states with large elementary trees; in turn, the sampling procedure is prone to local modes. In order to counteract this and to improve mixing we used simulated annealing. The probability mass function (Equations 5.5 - 5.6) was raised to the power $1/T$ with T dropping linearly from $T = 5$ to $T = 0$. Furthermore, using a final temperature of zero, we recover a maximum a posteriori (MAP) estimate which we denote $\mathbf{e}_{\text{MAP}}$.

Prediction

We discuss the problem of predicting a target tree $t_{N+1,t}$ that corresponds to a source tree $t_{N+1,s}$ unseen in the observed corpus $\mathbf{t}$. The maximum probability tree (MPT) can be found by considering all possible ways to derive it. However a much simpler alternative is to choose the target tree implied by the maximum probability derivation (MPD), which we define as

$$\begin{aligned} e^* &= \operatorname*{argmax}_e p(e \mid t_s, \mathbf{t}) \\ &= \operatorname*{argmax}_e \sum_{\mathbf{e}} p(e \mid t_s, \mathbf{e}) p(\mathbf{e} \mid \mathbf{t}) \end{aligned}$$

where e denotes a derivation for $t = t_s/t_t$. (We suppress the $N+1$ subscripts for brevity.) We approximate this objective first by substituting $\delta_{\mathbf{e}_{\text{MAP}}}(\mathbf{e})$ for $p(\mathbf{e} \mid \mathbf{t})$ and secondly using a finite STSG model for the infinite $p(e \mid t_s, \mathbf{e}_{\text{MAP}})$, which we obtain simply by normalizing the rule counts in $\mathbf{e}_{\text{MAP}}$. We use dynamic programming for parsing under this finite model (Eisner, 2003).[5]

Unfortunately, this approach does not ensure that the test instances are parsable, since t_s may include unseen structure or novel words. A work-around is to include all zero-count context free copy rules such as

$$\begin{aligned} \text{NP}/\text{NP} &\rightarrow (\text{NP NP}_{[1]}\ \text{PP}_{[2]})/(\text{NP NP}_{[1]}\ \text{PP}_{[2]}) \\ \text{NP}/\varepsilon &\rightarrow (\text{NP NP}_{[\varepsilon]}\ \text{PP}_{[\varepsilon]})/\varepsilon \end{aligned}$$

in order to smooth our finite model. We used Laplace smoothing (adding 1 to all counts) for simplicity.

[5] We experimented with MPT using Monte Carlo integration over possible derivations; the results were not significantly different from those using MPD.

5.1.2 Evaluation

We compared the Gibbs sampling compressor (GS) against a version of maximum a posteriori EM (with Dirichlet parameter greater than 1) and a discriminative STSG based on SVM training (Cohn and Lapata, 2008) (SVM). EM is a natural benchmark, while SVM is also appropriate since it can be taken as the state of the art for our task.[6]

We used a publicly available extractive sentence compression corpus: the Broadcast News compressions corpus (BNC) of Clarke and Lapata (2006a). This corpus consists of 1370 sentence pairs that were manually created from transcribed Broadcast News stories. We split the pairs into training, development, and testing sets of 1000, 170, and 200 pairs, respectively. The corpus was parsed using the Stanford parser (Klein and Manning, 2003b).

In our experiments with the publicly available SVM system we used all except paraphrasal rules extracted from bilingual corpora (Cohn and Lapata, 2008). The model chosen for testing had the parameter for trade-off between training error and margin set to $C = 0.001$, used margin rescaling, and Hamming distance over bags of tokens with brevity penalty for loss function. EM used a subset of the rules extracted by SVM, namely all rules except non-head deleting compression rules, and was initialized uniformly. Each EM instance was characterized by two parameters: α, the smoothing parameter for MAP-EM, and δ, the smoothing parameter for augmenting the learned grammar with rules

[6]The comparison system described by Cohn and Lapata (2008) attempts to solve a more general problem than ours, abstractive sentence compression. However, given the nature of the data that we provided, it can only learn to compress by deleting words. Since the system is less specialized to the task, their model requires additional heuristics in decoding not needed for extractive compression, which might cause a reduction in performance. Nonetheless, because the comparison system is a generalization of the extractive SVM compressor of Cohn and Lapata (2007), we do not expect that the results would differ qualitatively.

Table 5.1: Precision, recall, relational F1 and compression rate (%) for various systems on the 200-sentence BNC test set. The compression rate for the gold standard was 65.67%.

	SVM	EM	GS
Precision	55.60	58.80	58.94
Recall	53.37	56.58	64.59
Relational F1	54.46	57.67	61.64
Compression rate	59.72	64.11	65.52

Table 5.2: Average grammar and importance scores for various systems on the 20-sentence subsample. Scores marked with $*$ are significantly different than the corresponding GS score at $\alpha < .05$ and with † at $\alpha < .01$ according to post-hoc Tukey tests. ANOVA was significant at $p < .01$ both for grammar and importance.

	SVM	EM	GS	Gold
Grammar	$2.75^{\dagger}$	2.85^{*}	3.69	4.25
Importance	2.85	2.67^{*}	3.41	3.82
Comp. rate	68.18	64.07	67.97	62.34

extracted from unseen data (add-$(\delta - 1)$ smoothing was used), both of which were fit to the development set using grid-search over $(1, 2]$. The model chosen for testing was $(\alpha, \delta) = (1.0001, 1.01)$.

GS was initialized at a random derivation. We sampled the alignments of the source nodes in random order. The sampler was run for 5000 iterations with annealing. All hyperparameters α_c, β_c were held constant at α, β for simplicity and were fit using grid-search over $\alpha \in [10^{-6}, 10^{6}], \beta \in [10^{-3}, 0.5]$. The model chosen for testing was $(\alpha, \beta) = (100, 0.1)$.

As an automated metric of quality, we compute F-score based on grammatical relations (relational F1, or RelF1) (Riezler et al., 2003), by which the consistency between the set of predicted grammatical relations and those from the gold standard is measured, which has been shown by Clarke and Lapata (2006b) to correlate reliably with human judgments. We also conducted a small human subjective evaluation of the grammaticality and informativeness of the compressions generated by the various methods.

Automated evaluation

For all three systems we obtained predictions for the test set and used the Stanford parser to extract grammatical relations from predicted trees and the gold standard. We computed precision, recall, RelF1 (all based on grammatical relations), and compression rate (percentage of the words that are *retained*), which we report in Table 5.1. The results for GS are averages over five independent runs. EM gives a strong baseline since it already uses rules that are limited in depth and number of frontier nodes by stipulation, helping with the overfitting we have mentioned, surprisingly outperforming its discriminative counterpart in both precision and recall (and consequently RelF1). GS however maintains the same

level of precision as EM while improving recall, bringing an overall improvement in RelF1.

Human evaluation

We randomly subsampled our 200-sentence test set for 20 sentences to be evaluated by human judges through Amazon Mechanical Turk. We asked 15 self-reported native English speakers for their judgments of GS, EM, and SVM output sentences and the gold standard in terms of *grammaticality* (how fluent the compression is) and *importance* (how much of the meaning of and important information from the original sentence is retained) on a scale of 1 (worst) to 5 (best). We report in Table 5.2 the average scores. EM and SVM perform at very similar levels, which we attribute to using the same set of rules, while GS performs at a level substantially better than both, and much closer to human performance in both criteria. The human evaluation indicates that the superiority of the Bayesian nonparametric method is underappreciated by the automated evaluation metric.

The fact that GS performs better than EM can be attributed to two reasons: (1) GS uses a sparse prior and selects a compact representation of the data (grammar sizes ranged from 4K-7K for GS compared to a grammar of about 35K rules for EM). (2) GS does not commit to a precomputed grammar and searches over the space of all grammars to find one that bests represents the corpus. It is possible to introduce DP-like sparsity in EM using variational Bayes (VB) training (see Section 2.1.6 for an introduction). We experiment with this next in order to understand how dominant the two factors are. The VB algorithm requires a simple update to the M-step formulas for EM where the expected rule counts are normalized, such that instead of updating the rule weight in the t-th iteration as in the

Table 5.3: High probability ROOT / ROOT compression rules from the final state of the sampler.

(ROOT (S $CC_{[\varepsilon]}$ $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $NP_{[1]}$ $ADVP_{[\varepsilon]}$ $VP_{[2]}$ (. .)))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ (. .)))
(ROOT (S $ADVP_{[\varepsilon]}$ (, ,) $NP_{[1]}$ $VP_{[2]}$ (. .)))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ (. .)))
(ROOT (S $PP_{[\varepsilon]}$ (, ,) $NP_{[1]}$ $VP_{[2]}$ (. .)))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ (. .)))
(ROOT (S $PP_{[\varepsilon]}$ $,_{[\varepsilon]}$ $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $NP_{[\varepsilon]}$ (VP $VBP_{[\varepsilon]}$ (SBAR (S $NP_{[1]}$ $VP_{[2]}$))) $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $ADVP_{[\varepsilon]}$ $NP_{[1]}$ (VP $MD_{[2]}$ $VP_{[3]}$) $._{[4]}$))	/ (ROOT (S $NP_{[1]}$ (VP $MD_{[2]}$ $VP_{[3]}$) $._{[4]}$))
(ROOT (S (SBAR (IN as) $S_{[\varepsilon]}$) $,_{[\varepsilon]}$ $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $S_{[\varepsilon]}$ (, ,) $CC_{[\varepsilon]}$ (S $NP_{[1]}$ $VP_{[2]}$) $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $PP_{[\varepsilon]}$ $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $S_{[1]}$ (, ,) $CC_{[\varepsilon]}$ $S_{[2]}$ (. .)))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ (. .)))
(ROOT (S $S_{[\varepsilon]}$ $,_{[\varepsilon]}$ $NP_{[1]}$ $ADVP_{[2]}$ $VP_{[3]}$ $._{[4]}$))	/ (ROOT (S $NP_{[1]}$ $ADVP_{[2]}$ $VP_{[3]}$ $._{[4]}$))
(ROOT (S (NP (NP $NNP_{[\varepsilon]}$ (POS 's)) $NNP_{[1]}$ $NNP_{[2]}$) (VP (VBZ reports)) $._{[3]}$))	/ (ROOT (S (NP $NNP_{[1]}$ $NNP_{[2]}$) (VP (VBZ reports)) $._{[3]}$))
(ROOT (S $PP_{[1]}$ (, ,) $NP_{[2]}$ $ADVP_{[\varepsilon]}$ $VP_{[3]}$ $._{[4]}$))	/ (ROOT (S $PP_{[1]}$ (, ,) $NP_{[2]}$ $VP_{[3]}$ $._{[4]}$))
(ROOT (S $S_{[\varepsilon]}$ (, ,) $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))	/ (ROOT (S $NP_{[1]}$ $VP_{[2]}$ $._{[3]}$))
(ROOT (S $NP_{[\varepsilon]}$ (, ,) $PP_{[1]}$ (, ,) $NP_{[2]}$ $VP_{[3]}$ $._{[4]}$))	/ (ROOT (S $PP_{[1]}$ (, ,) $NP_{[2]}$ $VP_{[3]}$ $._{[4]}$))
(ROOT (NP (NP $DT_{[1]}$ (NN right)) $,_{[\varepsilon]}$ $NP_{[\varepsilon]}$ (. .)))	/ (ROOT (NP $DT_{[1]}$ (NN right) (. .)))
(ROOT (S (NP (PRP I)) (VP $VBP_{[\varepsilon]}$ (SBAR (IN that) $S_{[1]}$)) $._{[2]}$))	/ (ROOT (S (NP (DT that)) $VP_{[1]}$ $._{[2]}$))

Table 5.4: High probability S / S compression rules from the final state of the sampler.

(S NP$_{[1]}$ ADVP$_{[\varepsilon]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S INTJ$_{[\varepsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S (INTJ (UH Well)) ,$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S PP$_{[\varepsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S ADVP$_{[\varepsilon]}$ (, ,) S$_{[1]}$ (, ,) (CC but) S$_{[2]}$.$_{[3]}$)	/ (S S$_{[1]}$ (, ,) (CC but) S$_{[2]}$.$_{[3]}$)
(S ADVP$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S NP$_{[\varepsilon]}$ (VP VBP$_{[\varepsilon]}$ (SBAR (IN that) (S NP$_{[1]}$ VP$_{[2]}$))) (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\varepsilon]}$ (VP VBZ$_{[\varepsilon]}$ ADJP$_{[\varepsilon]}$ SBAR$_{[1]}$))	/ S$_{[1]}$
(S CC$_{[\varepsilon]}$ PP$_{[\varepsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\varepsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S NP$_{[1]}$ (, ,) ADVP$_{[\varepsilon]}$ (, ,) VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S CC$_{[\varepsilon]}$ (NP PRP$_{[1]}$) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(S ADVP$_{[\varepsilon]}$,$_{[\varepsilon]}$ PP$_{[\varepsilon]}$,$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S ADVP$_{[\varepsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S CC$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[\varepsilon]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S NP$_{[\varepsilon]}$ VP$_{[1]}$.$_{[2]}$)	/ (S VP$_{[1]}$.$_{[2]}$)
(S (PP IN$_{[1]}$ NP$_{[2]}$) NP$_{[\varepsilon]}$,$_{[3]}$ NP$_{[4]}$ VP$_{[5]}$ (. .))	/ (S (PP IN$_{[1]}$ NP$_{[2]}$) ,$_{[3]}$ NP$_{[4]}$ VP$_{[5]}$ (. .))
(S (CC but) ADVP$_{[\varepsilon]}$,$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S SBAR$_{[\varepsilon]}$,$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S PP$_{[\varepsilon]}$ (, ,) S$_{[1]}$,$_{[2]}$ CC$_{[3]}$ S$_{[4]}$ (. .))	/ (S S$_{[1]}$,$_{[2]}$ CC$_{[3]}$ S$_{[4]}$ (. .))
(S CC$_{[\varepsilon]}$,$_{[\varepsilon]}$ PP$_{[\varepsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S (, ,) PP$_{[\varepsilon]}$,$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S PP$_{[1]}$ NP$_{[\varepsilon]}$ (, ,) NP$_{[2]}$ VP$_{[3]}$.$_{[4]}$)	/ (S PP$_{[1]}$ (, ,) NP$_{[2]}$ VP$_{[3]}$.$_{[4]}$)
(S NP$_{[\varepsilon]}$ (VP VBD$_{[\varepsilon]}$ NP$_{[\varepsilon]}$ (SBAR IN$_{[\varepsilon]}$ (S NP$_{[1]}$ VP$_{[2]}$))) (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S (S NP$_{[1]}$ VP$_{[2]}$) (, ,) (CC and) (S NP$_{[\varepsilon]}$ VP$_{[3]}$) .$_{[4]}$)	/ (S NP$_{[1]}$ (VP VP$_{[2]}$ (, ,) (CC and) VP$_{[3]}$) .$_{[4]}$)
(S CC$_{[\varepsilon]}$ (, ,) SBAR$_{[\varepsilon]}$,$_{[\varepsilon]}$ NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\varepsilon]}$ (VP VBZ$_{[\varepsilon]}$ SBAR$_{[1]}$))	/ S$_{[1]}$
(S (NP DT$_{[\varepsilon]}$) (VP VBZ$_{[\varepsilon]}$ (SBAR (IN because) (S NP$_{[1]}$ VP$_{[2]}$))) (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S SBAR$_{[1]}$ (, ,) NP$_{[\varepsilon]}$ (VP VBZ$_{[\varepsilon]}$ (SBAR (S NP$_{[2]}$ VP$_{[3]}$))) (. .))	/ (S SBAR$_{[1]}$ (, ,) NP$_{[2]}$ VP$_{[3]}$ (. .))
(S NP$_{[1]}$ (, ,) ADVP$_{[\varepsilon]}$ (, ,) VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))

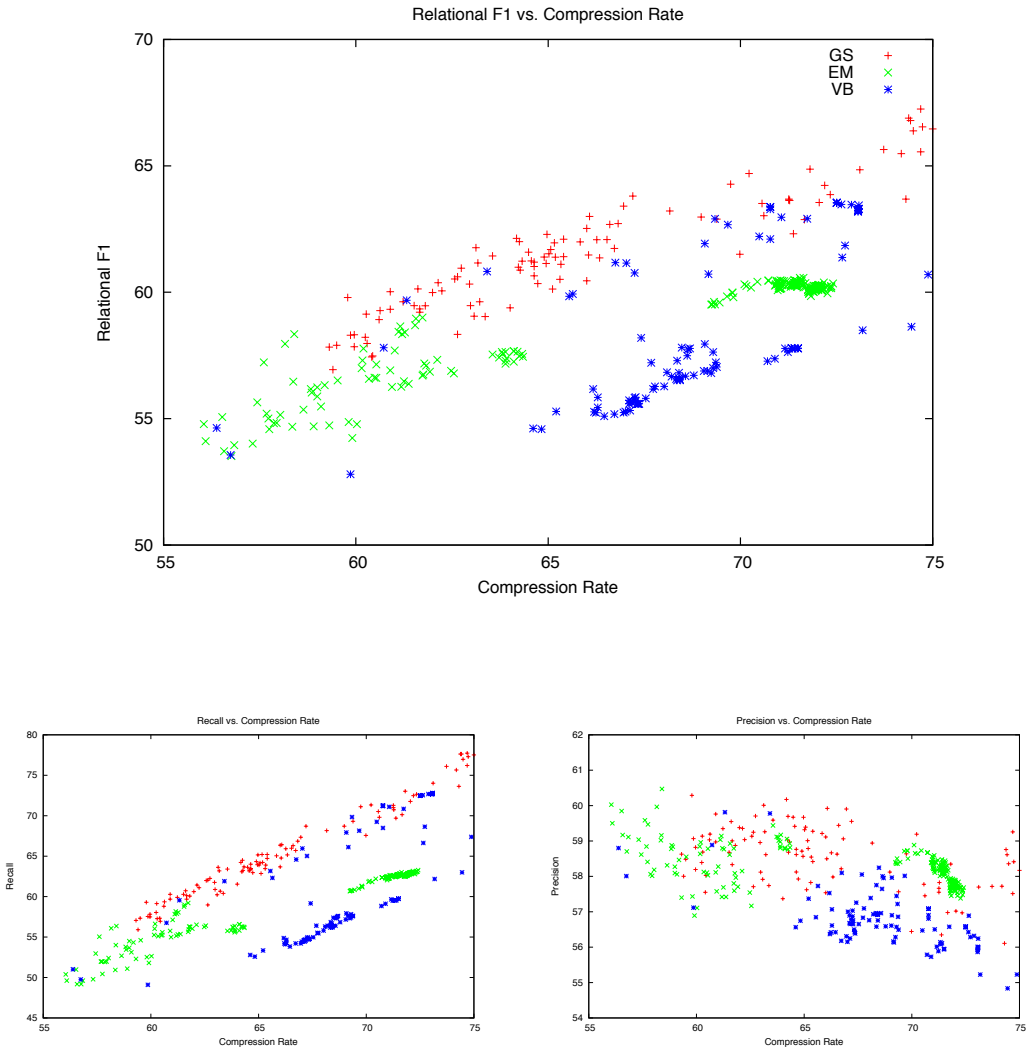


Figure 5.4: RefF1 (top), recall (bottom left), precision (bottom right) plotted against compression rate for GS, EM, VB. Higher values in the y-axes and smaller in the x are better, as this means good performance at a harsh compression rate.

following

$$\theta_{c,e}^{t+1} = \frac{n_{c,e} + \alpha - 1}{n_{c,\cdot} + K\alpha - K}$$

where $n_{c,e}$ represents the expected count of rule $c \to e$, and K is the total number of ways to rewrite c, we now take into account our DP$(\alpha_c, P_0(\cdot\,|\,c))$ prior in Equation 5.1, which, when truncated to a finite grammar, reduces to a K-dimensional Dirichlet prior with parameter $\alpha_c P_0(\cdot \mid c)$. Thus in VB we perform a *variational* E-step with the subprobabilities given by

$$\theta_{c,e}^{t+1} = \frac{\exp\left(\psi\left(n_{c,e} + \alpha_c P_0(e \mid c)\right)\right)}{\exp\left(\psi\left(n_{c,\cdot} + \alpha_c\right)\right)} \tag{5.7}$$

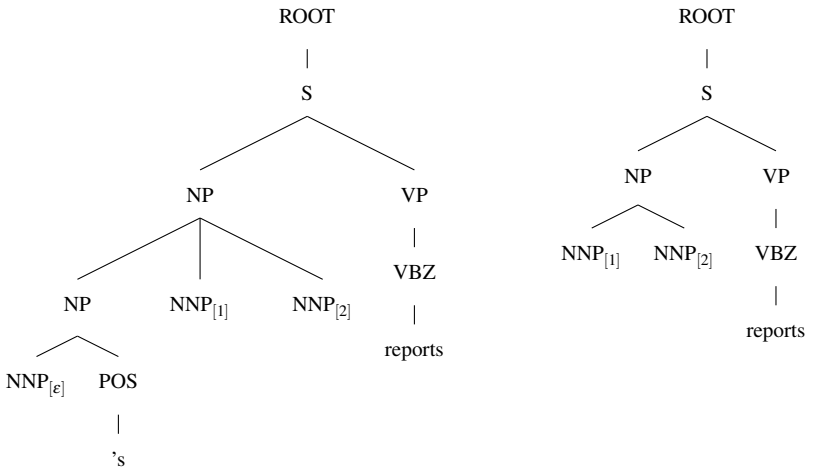
where $\psi(\cdot)$ denotes the digamma function (Liu and Gildea, 2009; MacKay, 1997). Hyper-parameters were handled the same way as for GS.

Instead of selecting a single model on the development set, here we provide the whole spectrum of models and their performances in order to better understand their comparative behavior. In Figure 5.4 we plot RelF1 on the test set versus compression rate and compare GS, EM, and VB ($\beta = 0.1$ fixed, (α, δ) ranging in $[10^{-6}, 10^{6}] \times (1, 2]$). Overall, we see that GS maintains roughly the same level of precision as EM (despite its larger compression rates) while achieving an improvement in recall, consequently performing at a higher RelF1 level. We note that VB somewhat bridges the gap between GS and EM, without quite reaching GS performance. We conclude that the mitigation of the two factors (narrowness and overfitting) both contribute to the performance gain of GS.[7]

In order to provide some insight into the grammar extracted by GS, we list in Tables 5.3 and 5.4 high probability subtree-deletion rules expanding categories ROOT / ROOT and S

[7] We have also experimented with VB with parametric independent symmetric Dirichlet priors. The results were similar to EM with the exception of sparse priors resulting in smaller grammars and slightly improving performance.

/ S, respectively. Of especial interest are deep lexicalized rules such as



a pattern of compression used many times in the BNC in sentence pairs such as "NPR's Anne Garrels reports" / "Anne Garrels reports". Such an informative rule with nontrivial collocation (between the possessive marker and the word "reports") would be hard to extract heuristically and can only be extracted by reasoning across the training examples.

5.1.3 Discussion

We explored nonparametric Bayesian learning of non-isomorphic tree mappings using Dirichlet process priors. We used the task of extractive sentence compression as a testbed to investigate the effects of sparse priors and nonparametric inference over the space of grammars. We showed that, despite its degeneracy, expectation maximization is a strong baseline when given a reasonable grammar. However, Gibbs-sampling–based nonparametric inference achieves improvements against this baseline. Our investigation with variational Bayes showed that the improvement is due both to finding sparse grammars (mitigating overfitting) and to searching over the space of all grammars (mitigating narrowness). Overall, we take these results as being encouraging for STSG induction via Bayesian nonpara-

metrics for monolingual translation tasks. The future for this work would involve natural extensions such as mixing over the space of word alignments; this would allow application to MT-like tasks where flexible word reordering is allowed, such as abstractive sentence compression and paraphrasing.

5.2 Further experiments with Web data

We have seen that the nonparametric Bayesian framework not only provides a unified and principled solution to model selection and estimation, but also achieves improved performance levels in the sentence compression test-bed. Our final claim in this thesis is that these capabilities of the framework (in particular the model selection capabilities) will truly shine in larger data set domains. Since the model selection process rests on re-use of patterns, larger data sets will allow us to induce compact yet informed grammars that provide good performance and rich linguistic analyses. To this end, in this section, we conduct a number of additional sentence compression experiments using the large compression data set that we collected from Wikipedia in Section 3.1. These experiments show that, similar to our result in Section 3.1, our Bayesian model trained on Wikipedia compressions has comparable performance even in out-of-domain settings. However, the benefit of the current model is that its linguistic sophistication is adaptive and determined via data-driven model selection as opposed to blind lexicalization. This provides not only interpretable but also compact grammar representations. We show some examples of our linguistically motivated compression patterns that we obtain from Wikipedia.

5.2.1 Experimental setup

For these experiments we use two main corpora: Broadcast News Corpus (BNC) from the previous section (1000 training pairs and 200 testing) and Wikipedia compressions that we previously used (for scalability reasons we use a subset of 100K pairs from the Wikipedia corpus). Since Wikipedia compressions are hard to predict (See Table 3.1 with compressions), we follow our previous approach in Section 3.1 of using these only as training data and performing all testing on the BNC compressions.

We run the Gibbs sampling compressor up to 1000 iterations on both datasets each, however this time instead of grid-searching over the hyperparameter space, we integrate these out via sampling. This gives us a parameter free training procedure. We followed the same Laplace smoothing CFG interpolation scheme as in the previous section. The RelF1 results (Table 5.5) are averages over 10 runs, the human evaluation (Table 5.6) is based on a single grammar for each dataset that had the best RelF1 on the development set.

5.2.2 Evaluation

The automatic evaluation in Table 5.5 shows BNC-trained STSG's power to closely predict its own test set; unfortunately, Wikipedia-trained STSG falls behind. We predict that this behavior is due to RelF1's sensitivity to the verbatim correspondence between the model predictions and the gold standard. In general, there are many ways to make valid compressions, even though a particular annotator for a data set may have a specific idea as to what makes a good compression. In order to test this point, we carry out a human evaluation. Following the same methodology as in the previous section we ask Amazon Mechanical Turk users to give grammaticality and importance ratings for each model and

Table 5.5: Average precision, recall, relational F1 and compression rate (%) for BNC-trained and Wikipedia-trained models on the BNC test set. The compression rate for the gold standard was 67.58%.

	BNC	Wikipedia
Precision	62.02	60.22
Recall	72.75	71.23
Relational F1	**66.95**	65.24
Compression rate	70.40	70.23

the gold standard on a scale of 1-5. In support of our prediction, Wikipedia-trained STSG catches up to BNC-trained STSG in human evaluation (see results in Table 5.6), attaining at least comparable performance. The fact that our Wikipedia-model can achieve this level of performance even when tested out-of-domain supports our claim of increased linguistic understanding of this task and thus better generality. Instead of purely memorizing the particularities of a given training set, our web-trained nonparametric inference learns natural rules that are naturally frequent in sentence compressions (see examples in Table 5.7).

5.2.3 Discussion

This result is similar to our lexicalized-SCFG result from Section 3.1. However, the benefit of the Bayesian nonparametric framework is evident from the grammar representations. Meanwhile the lexicalized-SCFG that we obtain from the 100K subsample of Wikipedia compressions is a sizable 960K-rule grammar, STSG provides a more compact

Table 5.6: Average grammar and importance scores for various systems on the BNC test set. Relational F1 is the score on development set based on which we choose our grammar to evaluate.

	BNC	Wikipedia	Gold
Grammar	3.68	**3.84**	4.23
Importance	**3.54**	3.51	3.95
Compression rate	71.11	69.94	67.58
Relational F1	68.40	66.93	100.00

representation of 15K rules.[8] Compact representations not only lend themselves to qualitative linguistic analysis, but also lead to faster parsing (compression) time by shrinking the grammar constant. Grammar sizes for the two models are plotted against training data set sizes in Figure 5.5.

Another advantage of the STSG approach is reduced estimation efforts. As we said, lexicalized-SCFG is an excessively parameterized model; therefore, significant smoothing efforts were needed to estimate these fine-grained probabilities (Collins, 2003). However, the rich-gets-richer dynamics of the Bayesian nonparametric STSG make sure that any found elementary tree patterns have robust representation in the data; therefore minimal smoothing (simple add-delta smoothing) was sufficient, only to make sure that the entire test set is parsable.

We claimed that the web-trained nonparametric STSG performs well out-of-domain by learning a compact set of patterns that are robust and descriptive of the task at hand and

[8]These numbers are directly comparable since our STSG can be flattened into an SCFG without loss of generality.

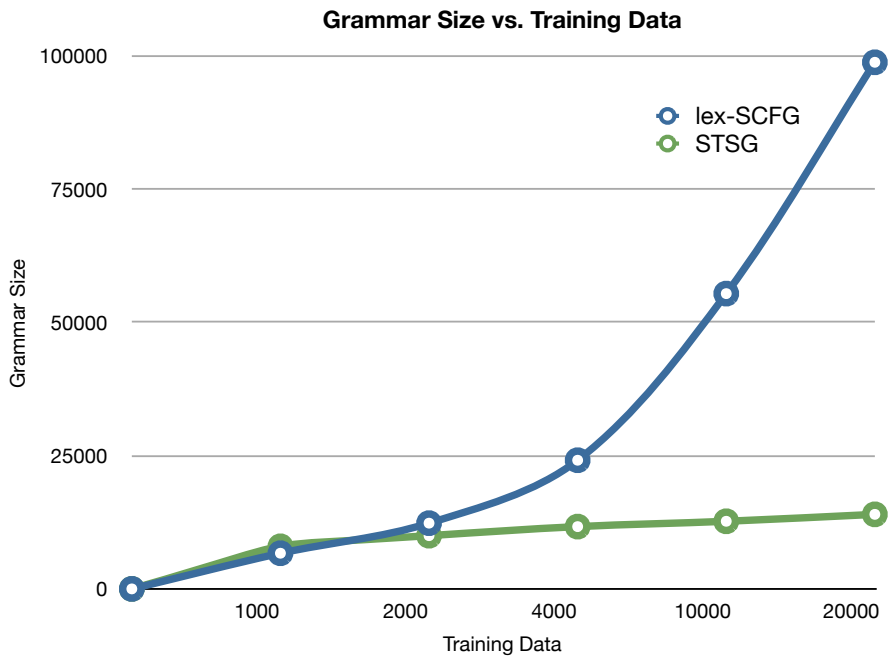


Figure 5.5: Grammar sizes for lexicalized-SCFG and STSG versus the amount of training data.

not the particularities of the data set creation process. In order to illustrate these patterns, in Table 5.7 we list the most frequent elementary tree pairs found in the training portion of the Wikipedia data. Our results support the claim that Bayesian nonparametric inference — under linguistically rich grammar formalisms when combined with large amounts of naturally occurring web-text — leads to effective and efficient systems, by providing good performance and compact representations, respectively.

Table 5.7: Example compression patterns from Wikipedia.

(NP NP$_{[1]}$ PP$_{[\varepsilon]}$)	/ NP$_{[1]}$
(NP (NP DT$_{[1]}$ JJ$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP DT$_{[1]}$ NN$_{[2]}$)
(NP NP$_{[\varepsilon]}$ PP$_{[1]}$)	/ PP$_{[1]}$
(NP (NP NP$_{[\varepsilon]}$ NNP$_{[1]}$) NNP$_{[2]}$)	/ (NP NNP$_{[1]}$ NNP$_{[2]}$)
(S (NP PRP$_{[\varepsilon]}$) VP$_{[1]}$)	/ VP$_{[1]}$
(NP NP$_{[\varepsilon]}$ (PP (IN of) NP$_{[1]}$))	/ NP$_{[1]}$
(NP NP$_{[\varepsilon]}$ NP$_{[1]}$)	/ NP$_{[1]}$
(NP (NP DT$_{[1]}$ JJ$_{[\varepsilon]}$) NNS$_{[2]}$)	/ (NP DT$_{[1]}$ NNS$_{[2]}$)
(NP NP$_{[1]}$ NN$_{[\varepsilon]}$)	/ NP$_{[1]}$
(NP NP$_{[1]}$,$_{[\varepsilon]}$)	/ NP$_{[1]}$
(SBAR (WHNP WDT$_{[\varepsilon]}$) (S VP$_{[1]}$))	/ VP$_{[1]}$
(S S$_{[\varepsilon]}$ S$_{[1]}$)	/ S$_{[1]}$
(NP (NP DT$_{[1]}$ JJ$_{[\varepsilon]}$) JJ$_{[2]}$)	/ (NP DT$_{[1]}$ JJ$_{[2]}$)
(NP (NP DT$_{[1]}$ JJ$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP DT$_{[1]}$ NN$_{[2]}$)
(S (NP NNP$_{[\varepsilon]}$ NNP$_{[1]}$) VP$_{[2]}$)	/ (S (NP NNP$_{[1]}$) VP$_{[2]}$)
(NP (NP NP$_{[\varepsilon]}$ NNP$_{[1]}$) NNP$_{[2]}$)	/ (NP NNP$_{[1]}$ NNP$_{[2]}$)
(VP VBP$_{[\varepsilon]}$ (SBAR S$_{[1]}$))	/ S$_{[1]}$
(S (S (NP PRP$_{[1]}$) (ADVP RB$_{[\varepsilon]}$)) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(VP VP$_{[1]}$ SBAR$_{[\varepsilon]}$)	/ VP$_{[1]}$
(NP (NP DT$_{[1]}$ NNP$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP DT$_{[1]}$ NN$_{[2]}$)
(NP NP$_{[1]}$ NN$_{[\varepsilon]}$)	/ NP$_{[1]}$
(S (S (S S$_{[\varepsilon]}$ (, ,)) CC$_{[\varepsilon]}$) S$_{[1]}$)	/ S$_{[1]}$
(S (S S$_{[\varepsilon]}$ (NP NNP$_{[1]}$)) VP$_{[2]}$)	/ (S (NP NNP$_{[1]}$) VP$_{[2]}$)
(PP IN$_{[1]}$ (NP JJ$_{[\varepsilon]}$ NN$_{[2]}$))	/ (PP IN$_{[1]}$ (NP NN$_{[2]}$))
(NP (NP PRP$$_{[1]}$ JJ$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP PRP$$_{[1]}$ NN$_{[2]}$)
(S CC$_{[\varepsilon]}$ NP$_{[1]}$)	/ NP$_{[1]}$
(S (VP TO$_{[\varepsilon]}$ VP$_{[1]}$))	/ VP$_{[1]}$
(VP VP$_{[1]}$ (ADVP RB$_{[\varepsilon]}$))	/ VP$_{[1]}$
(VP VP$_{[\varepsilon]}$ VP$_{[1]}$)	/ VP$_{[1]}$
(NP NP$_{[1]}$ JJ$_{[\varepsilon]}$)	/ NP$_{[1]}$
(VP VP$_{[1]}$,$_{[\varepsilon]}$)	/ VP$_{[1]}$
(NP (QP QP$_{[\varepsilon]}$ CD$_{[1]}$) NNS$_{[2]}$)	/ (NP CD$_{[1]}$ NNS$_{[2]}$)
(NP (NP DT$_{[1]}$ JJ$_{[2]}$) NN$_{[\varepsilon]}$)	/ (NP DT$_{[1]}$ JJ$_{[2]}$)
(VP VBZ$_{[1]}$ (ADJP RB$_{[\varepsilon]}$ JJ$_{[2]}$))	/ (VP VBZ$_{[1]}$ (ADJP JJ$_{[2]}$))
(NP (NP DT$_{[\varepsilon]}$ NNS$_{[1]}$) PP$_{[2]}$)	/ (NP (NP NNS$_{[1]}$) PP$_{[2]}$)
(VP (VP MD$_{[1]}$ (ADVP RB$_{[\varepsilon]}$)) VP$_{[2]}$)	/ (VP MD$_{[1]}$ VP$_{[2]}$)
(S CC$_{[\varepsilon]}$ (PP IN$_{[1]}$ NP$_{[2]}$))	/ (PP IN$_{[1]}$ NP$_{[2]}$)
(VP VP$_{[\varepsilon]}$ PP$_{[1]}$)	/ PP$_{[1]}$
(S (S S$_{[\varepsilon]}$ (NP PRP$_{[1]}$)) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(S (S (NP PRP$_{[1]}$) (ADVP RB$_{[\varepsilon]}$)) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(VP VP$_{[1]}$ NP$_{[\varepsilon]}$)	/ VP$_{[1]}$
(NP (NP (DT the) JJ$_{[\varepsilon]}$) NN$_{[1]}$)	/ (NP (DT the) NNP$_{[1]}$)
(VP VP$_{[\varepsilon]}$ SBAR$_{[1]}$)	/ S$_{[1]}$
(S S$_{[1]}$ S$_{[\varepsilon]}$)	/ S$_{[1]}$
(NP (NP DT$_{[1]}$ NN$_{[\varepsilon]}$) NNS$_{[2]}$)	/ (NP DT$_{[1]}$ NNS$_{[2]}$)
(S (S S$_{[\varepsilon]}$ (NP EX$_{[1]}$)) VP$_{[2]}$)	/ (S (NP EX$_{[1]}$) VP$_{[2]}$)
(S S$_{[\varepsilon]}$ NP$_{[1]}$)	/ NP$_{[1]}$
(VP VP$_{[1]}$ PP$_{[\varepsilon]}$)	/ VP$_{[1]}$
(PP IN$_{[\varepsilon]}$ NP$_{[1]}$)	/ NP$_{[1]}$
(NP (NP (DT a) JJ$_{[\varepsilon]}$) NN$_{[1]}$)	/ (NP (DT a) NN$_{[1]}$)
(S (S S$_{[\varepsilon]}$ (NP PRP$_{[1]}$)) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(VP VP$_{[1]}$ (ADVP RB$_{[\varepsilon]}$))	/ VP$_{[1]}$
(SBAR (S (NP PRP$_{[\varepsilon]}$) VP$_{[1]}$))	/ VP$_{[1]}$
(NP NP$_{[\varepsilon]}$ PP$_{[1]}$)	/ NP$_{[1]}$
(S NP$_{[1]}$ (ADVP RB$_{[\varepsilon]}$))	/ NP$_{[1]}$
(SBAR IN$_{[\varepsilon]}$ S$_{[1]}$)	/ S$_{[1]}$
(NP (NP DT$_{[1]}$ NN$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP DT$_{[1]}$ NN$_{[2]}$)
(NP (NP DT$_{[\varepsilon]}$ NN$_{[1]}$) PP$_{[2]}$)	/ (NP (NP NN$_{[1]}$) PP$_{[2]}$)
(NP (NP NP$_{[1]}$ NN$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP NP$_{[1]}$ NN$_{[2]}$)
(PP IN$_{[\varepsilon]}$ NP$_{[1]}$)	/ PP$_{[1]}$
(VP (VP VBP$_{[1]}$ (ADVP RB$_{[\varepsilon]}$)) VP$_{[2]}$)	/ (VP VBP$_{[1]}$ VP$_{[2]}$)
(VP VBP$_{[\varepsilon]}$ SBAR$_{[1]}$)	/ S$_{[1]}$
(NP NP$_{[1]}$ JJ$_{[\varepsilon]}$)	/ NP$_{[1]}$
(NP (NP NP$_{[1]}$ (CC and)) NP$_{[\varepsilon]}$)	/ NP$_{[1]}$
(NP NP$_{[\varepsilon]}$ SBAR$_{[1]}$)	/ SBAR$_{[1]}$
(S S$_{[\varepsilon]}$ VP$_{[1]}$)	/ S$_{[1]}$
(S S$_{[1]}$ (NP PRP$_{[\varepsilon]}$))	/ S$_{[1]}$
(VP VP$_{[1]}$ PP$_{[\varepsilon]}$)	/ VP$_{[1]}$
(NP (NP NN$_{[\varepsilon]}$) (PP (IN of) NP$_{[1]}$))	/ NP$_{[1]}$
(VP VP$_{[1]}$ (S VP$_{[\varepsilon]}$))	/ VP$_{[1]}$
(PP (PP (ADVP RB$_{[\varepsilon]}$) IN$_{[1]}$) NP$_{[2]}$)	/ (PP IN$_{[1]}$ NP$_{[2]}$)
(NP (NP NP$_{[1]}$,$_{[\varepsilon]}$) NP$_{[\varepsilon]}$)	/ NP$_{[1]}$
(S (ADVP RB$_{[\varepsilon]}$) NP$_{[1]}$)	/ NP$_{[1]}$
(S (VP (TO to) (VP VP$_{[1]}$ PP$_{[\varepsilon]}$)))	/ (S (VP (TO to) VP$_{[1]}$))
(S (VP VBD$_{[\varepsilon]}$ VP$_{[1]}$))	/ VP$_{[1]}$
(NP NP$_{[1]}$ SBAR$_{[\varepsilon]}$)	/ NP$_{[1]}$
(NP NP$_{[1]}$ (SBAR (WHNP WDT$_{[\varepsilon]}$) (S VP$_{[\varepsilon]}$)))	/ NP$_{[1]}$
(VP (VP (VBZ 's) (ADVP RB$_{[\varepsilon]}$)) VP$_{[1]}$)	/ (VP (VBZ 's) VP$_{[1]}$)
(S S$_{[1]}$ (ADVP RB$_{[\varepsilon]}$))	/ S$_{[1]}$
(VP (VP (VBZ has) (ADVP RB$_{[\varepsilon]}$)) VP$_{[1]}$)	/ (VP (VBZ has) VP$_{[1]}$)
(NP (QP DT$_{[\varepsilon]}$ CD$_{[1]}$) NNS$_{[2]}$)	/ (NP CD$_{[1]}$ NNS$_{[2]}$)
(NP (NP PRP$$_{[1]}$ JJ$_{[\varepsilon]}$) NNS$_{[2]}$)	/ (NP PRP$$_{[1]}$ NNS$_{[2]}$)
(VP VP$_{[1]}$ (ADVP RB$_{[\varepsilon]}$))	/ VP$_{[1]}$
(NP (NP NP$_{[1]}$,$_{[\varepsilon]}$) NP$_{[\varepsilon]}$)	/ NP$_{[1]}$
(S (S CC$_{[\varepsilon]}$ (NP PRP$_{[1]}$)) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(SBAR IN$_{[\varepsilon]}$ S$_{[1]}$)	/ VP$_{[1]}$
(VP (VP (VBZ is) (ADVP RB$_{[\varepsilon]}$)) VP$_{[1]}$)	/ (VP (VBZ is) VP$_{[1]}$)
(VP (VP VP$_{[1]}$ (CC and)) VP$_{[\varepsilon]}$)	/ VP$_{[1]}$
(VP (VP VBZ$_{[1]}$ (ADVP RB$_{[\varepsilon]}$)) ADJP$_{[2]}$)	/ (VP VBZ$_{[1]}$ ADJP$_{[2]}$)
(S S$_{[\varepsilon]}$ PP$_{[1]}$)	/ PP$_{[1]}$
(S S$_{[1]}$ (ADVP RB$_{[\varepsilon]}$))	/ NP$_{[1]}$
(NP (NP DT$_{[1]}$ VBG$_{[\varepsilon]}$) NN$_{[2]}$)	/ (NP DT$_{[1]}$ NN$_{[2]}$)

Chapter 6

Conclusion

In this thesis, we argued that Web data not only serves to improve the performance of simple models, but also can allow the use of qualitatively more sophisticated models that would not be deployable otherwise, leading to even further performance boosts. We first investigated this hypothesis through the use of parametric techniques, by showing improvements when using models of finer granularity in Web-scale sentence compression and lexical correction domains. In our attempt to answer the question of how to automatically determine the optimal level of granularity of a model given a dataset, we developed a portfolio of Bayesian nonparametric grammar induction techniques for linguistically sophisticated grammar formalisms and achieved quantitative and qualitative improvements in the domain of syntactic parsing. This portfolio included the first ever principled probabilistic model selection method for TAG as well as an approximate parsing solution that scales as fast as context-free formalisms. Finally we tested our nonparametric grammar induction scheme on Web-scale sentence compression experiments and showed quantitative improvements against simple models and qualitative improvements in compactness against

rich parametric models.

There is always room for further improvement in grammar induction. Some straightforward extensions of our work include grammar refinement through hierarchical nonparametric priors (Shindo et al., 2012; Liang et al.), more efficient or effective inference schemes such as parallel MCMC (Williamson et al., 2013) and fast variational online inference (Hoffman et al., 2010). Currently, our grammar induction system can reasonably scale up to hundreds of thousands of examples. In order to process the entirety of the millions of examples found in Wikipedia or all over the Web, there will be need for ever faster and leaner solutions.

Although in this thesis we narrowed in on Wikipedia revisions and the particular application domains of sentence compression and lexical correction, the arguments and methods discussed herein are widely applicable to general textual editing and our vision has ramifications beyond Wikipedia itself. Web technologies can benefit from samples of human text editing behavior occurring in abundance throughout Web 2.0 platforms for general text editing tasks such as paraphrasing, grammatical correction, translation, and so on. Today's Web is created by people. In every step of the process by which the Web is created and evolved for the better, human annotation is involved, a potentially valuable signal that is currently mostly lost. We see a future for the Web in which computers increasingly help create the Web, and we envision a cycle of feedback between computers and humans by which not only do the computers learn to mimic human behavior, but also humans learn to steer the computers in the right direction through the ways in which they interact with the Web, enabling a more organic relationship to form.

An example of this cycle of feedback has already been happening in Web search: Search

engines adapt to human behavior by learning from how users formulate their queries in particular contexts and navigate the results (Kamvar and Baluja, 2007, 2008). At the same time, in order to better navigate these systems, humans learn ways to steer the engines (for example, they have learned to query by using few high-density keywords) effectively training them through their searching behavior. Perhaps even more directly, von Ahn et al. (2008) used the human annotation produced by the use of CAPTCHAs for Web security to improve optical character recognition and to help digitize large collections of printed material. Similarly, in language education and machine translation, crowd-sourcing solutions has proven to be hugely effective for data collection as well as improving user experience (von Ahn and Dabbish, 2004; von Ahn, 2013). We believe that similar kinds of signals exist and can be useful in many more NLP tasks such as paraphrasing, grammar correction, summarization, and generation, if every human interaction is seen as valuable data and folded back into the system through machine learning.

Bibliography

Eneko Agirre and Philip Edmonds, editors. *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*. Springer, 2006.

Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219914.

Regina Barzilay and Lillian Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 16–23, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073448.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007. ISBN 0387310738. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387310738`.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. doi: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL `http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993`.

Rens Bod. Towards a Formal Language Theory of Stochastic Grammars. In *Beyond Grammar: An Experience-based Theory of Language*, CSLI Lecture notes number 88, chapter 3, pages 30–44. CSLI Publications, Stanford, 1998.

Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *ACL*, 2000.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990. ISSN 0891-2017.

Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47, March 2006.

Andrew J. Carlson, Jeffrey Rosen, and Dan Roth. Scaling up context sensitive text correction. In *IAAI2001*, pages 45–50, 2001.

Xavier Carreras, Michael Collins, and Terry Koo. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 9–16, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-48-4. URL `http://dl.acm.org/citation.cfm?id=1596324.1596327`.

Jonathan Chang, Jordan Boyd-Graber, and David M. Blei. Connections between the lines: augmenting social networks with text. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: http://doi.acm.org/10.1145/1557019.1557044.

Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, AAAI'97/IAAI'97, pages 598–603. AAAI Press, 1997. ISBN 0-262-51095-2. URL `http://portal.acm.org/citation.cfm?id=1867406.1867499`.

David Chiang. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 456–463, Morristown, NJ, USA, 2000. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1075218.1075276. URL `http://dx.doi.org/10.3115/1075218.1075276`.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219873.

James Clarke and Mirella Lapata. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 144–151, Sydney, Australia, July 2006a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P06/P06-2019`.

James Clarke and Mirella Lapata. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 377–384, Sydney, Australia, July 2006b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P06/P06-1048`.

Trevor Cohn and Phil Blunsom. A Bayesian model of syntax-directed tree to string grammar induction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 352–361, Morristown, NJ, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6.

Trevor Cohn and Phil Blunsom. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 225–230, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1858842.1858884`.

Trevor Cohn and Mirella Lapata. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning*, pages 73–82, Prague, 2007. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 137–144, Manchester, United Kingdom, 2008. Association for Computational Linguistics.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. Inducing compact but accurate tree-substitution grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Morristown, NJ, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1.

Michael Collins. Three generative, lexicalized models for statistical parsing. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Somerset, New Jersey, 1997. Association for Computational Linguistics. URL `citeseer.ist.psu.edu/article/collins97three.html`.

Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003. ISSN 0891-2017.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, 2005.

Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D07/D07-1074`.

Hal Daumé, Kevin Knight, Irene Langkilde-Geary, Daniel Marcu, and Kenji Yamada. The importance of lexicalized syntax models for natural language generation tasks. Proceedings of the Second International Conference on Natural Language Generation. Arden House, NJ, July 1-3, 2002.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B): 1–38, 1977.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. Why generative phrase models underperform surface heuristics. In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 31–38, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. Sampling alignment structure under a Bayesian translation model. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

Christine Doran, Beth Hockey, Philip Hopely, Joseph Rosenzweig, Anoop Sarkar, B. Srinivas, Fei Xia, Alexis Nasr, and Owen Rambow. Maintaining the forest and burning out the underbrush in XTAG. In *Proceedings of the ENVGRAM Workshop*, 1997.

Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 205–208, Morristown, NJ, USA, 2003. Association for Computational Linguistics. ISBN 0-111-456789. doi: http://dx.doi.org/10.3115/1075178.1075217.

Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

Michel Galley and Kathleen McKeown. Lexicalized Markov grammars for sentence compression. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 180–187, Rochester, New York, April 2007. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL `http://www1.cs.columbia.edu/~galley/papers/syntax-mt.pdf`.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, 6(6):721–741, 1984. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.1984.4767596.

Daniel Gildea. Loosely tree-based alignment for machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 80–87, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1075096.1075107.

Michaela Goetz, Jure Leskovec, Mary Mcglohon, and Christos Faloutsos. Modeling blog dynamics. In *International Conference on Weblogs and Social Media*, May 2009.

Sharon Goldwater and Thomas Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P07-1094`.

Sharon Goldwater, Thomas Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P06/P06-1085`.

Joshua T. Goodman. *Parsing inside-out*. PhD thesis, Harvard University, Cambridge, MA, USA, 1998. AAI9832377.

Joshua T. Goodman. Efficient parsing of DOP with PCFG-reductions. *Bod et al. 2003*, 2002. URL `http://research.microsoft.com/~joshuago/dop-csli.ps`.

Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009. ISSN 1541-1672. doi: http://doi.ieeecomputersociety.org/10.1109/MIS.2009.36. URL `http://www.computer.org/portal/cms_docs_intelligent/intelligent/homepage/2009/x2exp.pdf`.

David Hall, Daniel Jurafsky, and Christopher D. Manning. Studying the history of ideas using topic models. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 363–371, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. ISSN 1464-3510. doi: 10.1093/biomet/57.1.97. URL `http://dx.doi.org/10.1093/biomet/57.1.97`.

Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111, March 2005.

Matthew D. Hoffman, David M. Blei, and Francis R. Bach. Online learning for latent dirichlet allocation. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *NIPS*, pages 856–864. Curran Associates, Inc., 2010. URL http://dblp.uni-trier.de/db/conf/nips/nips2010.html#HoffmanBB10.

Rebecca Hwa. An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *Proceedings of the 17th international conference on Computational linguistics - Volume 1*, pages 557–563, Morristown, NJ, USA, 1998. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/980451.980938. URL http://dx.doi.org/10.3115/980451.980938.

Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics, ROCLING X*, Taiwan, 1998.

Hongyan Jing. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315, Morristown, NJ, USA, 2000. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/974147.974190.

Mark Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1031.

Mark Johnson and Thomas Griffiths. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of the North American Conference on Computational Linguistics (NAACL '07)*, 2007.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N07/N07-1018.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, November 1999. ISSN 0885-6125. doi: 10.1023/A:1007665907178. URL http://dx.doi.org/10.1023/A:1007665907178.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.

Maryam Kamvar and Shumeet Baluja. Deciphering trends in mobile search. *Computer*, 40(8):58–62, August 2007. ISSN 0018-9162. doi: 10.1109/MC.2007.270. URL `http://dx.doi.org/10.1109/MC.2007.270`.

Maryam Kamvar and Shumeet Baluja. Query suggestions for mobile search: understanding usage patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1013–1016, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357210. URL `http://doi.acm.org/10.1145/1357054.1357210`.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003a. Association for Computational Linguistics. doi: 10.3115/1075096.1075150. URL `http://dx.doi.org/10.3115/1075096.1075150`.

Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS*, pages 3–10. MIT Press, 2003b.

Kevin Knight and Daniel Marcu. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107, 2002. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(02)00222-9.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073462.

Irene Langkilde. Forest-based statistical sentence generation. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 170–177, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Mirella Lapata and Frank Keller. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):3, 2005. ISSN 1550-4875. doi: http://doi.acm.org/10.1145/1075389.1075392.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 915–924, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367620.

Jure Leskovec, Lars Backstrom, and Jon Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 497–506, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: http://doi.acm.org/10.1145/1557019.1557077.

Percy Liang, Slav Petrov, Michael I. Jordan, and Dan Klein. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP-CoNLL*, pages 688–697. ACL.

Ding Liu and Daniel Gildea. Bayesian learning of phrasal tree-to-string templates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1308–1317, Singapore, August 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D09/D09-1136`.

Edward Loper and Steven Bird. NLTK: The natural language toolkit, 2002. URL `citeseer.ist.psu.edu/loper02nltk.html`.

David J.C. MacKay. Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge, UK, 1997.

David M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283, Morristown, NJ, USA, 1995. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/981658.981695. URL `http://dx.doi.org/10.3115/981658.981695`.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330, June 1993. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972470.972475`.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. Context based spelling correction. *Information Processing and Management*, 27(5):517–522, 1991. ISSN 0306-4573. doi: http://dx.doi.org/10.1016/0306-4573(91)90066-U.

Andrew McCallum, Xuerui Wang, and Andrés Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *Journal of Artificial Intelligence Research*, 30(1):249–272, 2007. ISSN 1076-9757.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. ISSN 00219606. doi: 10.1063/1.1699114. URL http://dx.doi.org/10.1063/1.1699114.

Rada Mihalcea. Using Wikipedia for automatic word sense disambiguation. In *HLT-NAACL*, pages 196–203, 2007.

Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

Rani Nelken and Elif Yamangil. Mining Wikipedia's article revision history for training computational linguistics algorithms. In *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy (WikiAI '08)*. Association for the Advancement of Artificial Intelligence, 2008.

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004. ISSN 0891-2017. doi: http://dx.doi.org/10.1162/0891201042544884.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1614025.1614037.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Morristown, NJ, USA, 2006. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1220175.1220230. URL http://dx.doi.org/10.3115/1220175.1220230.

Matt Post and Daniel Gildea. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-2012.

Detlef Prescher, Remko Scha, Khalil Sima'an, and Andreas Zollmann. On the statistical consistency of DOP estimators. In Bart Decadt, Vronique Hoste, and Guy De Pauw, editors, *CLIN*, volume 111 of *Antwerp papers in linguistics*. University of Antwerp, 2003. URL http://dblp.uni-trier.de/db/conf/clin/clin2003.html#PrescherSSZ03.

Chris Quirk, Chris Brockett, and William Dolan. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149, 2004.

Philip Resnik. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 418–424, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. doi: 10.3115/992133.992135. URL `http://dx.doi.org/10.3115/992133.992135`.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 118–125, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073471.

Herbert Rubenstein and John Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

Remko Scha and Rens Bod. Efficient parsing of DOP with PCFG-reductions, October 2003. URL `http://citeseer.ist.psu.edu/661856.html;http://turing.wins.uva.nl/~rens/dopvolume/bodschasimaan.ps`.

Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *COLING-92*, pages 426–433, Nantes, France, 1992.

Yves Schabes and Stuart M. Shieber. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124, 1994. Also available as cmp-lg/9404001.

Yves Schabes and Richard C. Waters. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21:479–513, December 1995. ISSN 0891-2017. URL `http://portal.acm.org/citation.cfm?id=218355.218363`.

Stuart M. Shieber. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In Dekai Wu and David Chiang, editors, *Proceedings of the Workshop on Syntax and Structure in Statistical Translation*, Rochester, New York, 26 April 2007. URL `http://www.eecs.harvard.edu/~shieber/Biblio/Papers/stag-mt-basis.pdf`.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing. *J. Log. Program.*, 24(1&2):3–36, 1995.

Hiroyuki Shindo, Akinori Fujino, and Masaaki Nagata. Insertion operator for Bayesian tree substitution grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 206–211, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL `http://dl.acm.org/citation.cfm?id=2002736.2002780`.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–448, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-1046`.

Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using Wikipedia. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, pages 1419–1424. Menlo Park, CA; Cambridge, MA; London; AAAI Press, 2006.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. URL `http://www.gatsby.ucl.ac.uk/~ywteh/research/npbayes/jasa2006.pdf`.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005. ISSN 1533-7928.

Jenine Turner and Eugene Charniak. Supervised and unsupervised learning for sentence compression. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 290–297, Morristown, NJ, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219876.

Jurgen Van Gael, Andreas Vlachos, and Zoubin Ghahramani. The infinite HMM for unsupervised POS tagging. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, EMNLP '09, pages 678–687, Morristown, NJ, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-62-6. URL `http://portal.acm.org/citation.cfm?id=1699571.1699601`.

Fernanda B. Viégas, Martin Wattenberg, and Kushal Dave. Studying cooperation and conflict between authors with *history flow* visualizations. In Elizabeth Dykstra-Erickson and Manfred Tscheligi, editors, *CHI*, pages 575–582. ACM, 2004. ISBN 1-58113-702-8.

Luis von Ahn. Duolingo: learn a language for free while helping to translate the web. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, IUI '13,

pages 1–2, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1965-2. doi: 10.1145/2449396.2449398. URL http://doi.acm.org/10.1145/2449396.2449398.

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: 10.1145/985692.985733. URL http://doi.acm.org/10.1145/985692.985733.

Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008. doi: 10.1126/science.1160379. URL http://www.sciencemag.org/content/321/5895/1465.abstract.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1003.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Ged Ellis. Using the Web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D09/D09-1093.

Sinead Williamson, Avinava Dubey, and Eric P. Xing. Parallel Markov chain Monte Carlo for nonparametric mixture models. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 98–106. JMLR Workshop and Conference Proceedings, 2013. URL http://jmlr.csail.mit.edu/proceedings/papers/v28/williamson13.pdf.

I. Witten and T. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4), 1991.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997. ISSN 0891-2017.

Fei Xia, Chung-hye Han, Martha Palmer, and Aravind Joshi. Automatically extracting and comparing lexicalized grammars for different languages. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, IJCAI'01, pages 1321–1326, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-812-5, 978-1-558-60812-2. URL http://dl.acm.org/citation.cfm?id=1642194.1642271.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530, Morristown, NJ, USA, 2001. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073012.1073079.

Elif Yamangil and Rani Nelken. Mining Wikipedia revision histories for improving sentence compression. In *Proceedings of ACL-08: HLT, Short Papers*, pages 137–140, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-2035`.

Elif Yamangil and Stuart Shieber. Bayesian synchronous tree-substitution grammar induction and its application to sentence compression. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 937–947, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P10-1096`.

Elif Yamangil and Stuart Shieber. Estimating compact yet rich tree insertion grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-2022`.

Elif Yamangil and Stuart Shieber. Nonparametric Bayesian inference and cubic-time parsing for tree-adjoining grammars. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sophia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P12-2022`.

Appendix A

Variational Inference for GMM

Let us look at the deep GMM where $\pi \sim \text{Dirichlet}(\alpha)$ and infer π and Z via variational inference. The log-joint-likelihood has an undesirable coupling between these two sets of variables π and Z as they appear in a product together.

$$\begin{aligned}
\log P(X,Z,\pi|\mu,\sigma^2,\alpha) &= \log\left\{\prod_{k=1}^{K}\pi_k^{\alpha_k-1}\prod_{n=1}^{N}\left[\prod_{k=1}^{K}\pi_k^{z_{nk}}\prod_{k=1}^{K}\text{N}(x_n \mid \mu_k,\sigma^2)^{z_{nk}}\right]\right\} \\
&\quad + \text{constants} \\
&= \sum_{k=1}^{K}(\alpha_k-1)\log\pi_k + \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\log\pi_k \;\leftarrow \text{coupling} \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\log\text{N}(x_n \mid \mu_k,\sigma^2)
\end{aligned}$$

In order to break the coupling, we assume *factorized* variational families

$$\begin{aligned}
Q(\pi,Z) &= Q(\pi)\prod_{n=1}^{N}Q(z_n) \\
&= \text{Dirichlet}(\pi|\gamma)\prod_{n=1}^{N}\text{Multinomial}(z_n|\phi_n) \\
&= \text{C}(\gamma)\prod_{k=1}^{K}\pi_k^{\gamma_k-1}\prod_{n=1}^{N}\prod_{k=1}^{K}\phi_{nk}^{z_{nk}}
\end{aligned}$$

with variational parameters $\gamma = \{\gamma_k\}, \phi = \{\phi_{nk}\}$ where $\mathrm{C}(\gamma) = \Gamma(\sum \gamma_k) / \prod_k \Gamma(\gamma_k)$ is the normalization constant of the Dirichlet distribution. The variational lower bound can be written as follows

$$
\begin{aligned}
\log P(X|\mu,\sigma^2,\alpha) &\geq \mathbb{L}[\,\gamma,\phi\,] \\
&= \mathbb{E}_{Q(\pi,Z|\gamma,\phi)}\big[\log P(X,Z,\pi|\mu,\sigma^2,\alpha)\big] \\
&\quad + \mathbb{H}[Q(\pi,Z|\gamma,\phi)] \\
&= \sum_{k=1}^{K}(\alpha_k-1)\mathbb{E}_Q[\log\pi_k] + \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Q[z_{nk}]\,\mathbb{E}_Q[\log\pi_k] \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Q[z_{nk}]\log \mathrm{N}(x_n \mid \mu_k,\sigma^2) \\
&\quad - \log\Gamma(\sum_{k=1}^{K}\gamma_k) + \sum_{k=1}^{K}\log\Gamma(\gamma_k) - \sum_{k=1}^{K}(\gamma_k-1)\mathbb{E}_Q[\log\pi_k] \\
&\quad - \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Q[z_{nk}]\log\phi_{nk} \\
&= \sum_{k=1}^{K}(\alpha_k-1)\tilde{\psi}_k + \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_{nk}\tilde{\psi}_k + \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_{nk}\log\mathrm{N}(x_n \mid \mu_k,\sigma^2) \\
&\quad - \log\Gamma(\sum_{k=1}^{K}\gamma_k) + \sum_{k=1}^{K}\log\Gamma(\gamma_k) - \sum_{k=1}^{K}(\gamma_k-1)\tilde{\psi}_k \\
&\quad - \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_{nk}\log\phi_{nk}
\end{aligned}
$$

where we have introduced the *Digamma function* ψ the derivative of the log-Γ function, and defined $\tilde{\psi}_k = \mathbb{E}_Q[\log\pi_k] = \psi(\gamma_k) - \psi(\sum_k \gamma_k)$. Now let us first maximize the lower bound with respect to γ (the variational distribution of π) by collecting all terms containing γ_k:

$$
\mathbb{L}[\,\gamma_k\,] = \sum_{k=1}^{K}\delta_k\tilde{\psi}_k - \log\Gamma(\sum_{k=1}^{K}\gamma_k) + \log\Gamma(\gamma_k)
$$

(define $\delta_k = \sum_{n=1}^{N} \phi_{nk} + \alpha_k - \gamma_k$ to simplify notation) and taking derivative with respect to γ_k,

$$\begin{aligned}\frac{\partial}{\partial \gamma_k} \mathbb{L}[\gamma_k] &= -\psi(\gamma_k) + \delta_k \psi'(\gamma_k) - \psi'(\sum_k \gamma_k) \sum_k \delta_k + \psi(\sum_k \gamma_k) \\ &\quad - \psi(\sum_k \gamma_k) + \psi(\gamma_k)\end{aligned}$$

and setting it equal to zero we have $\delta_k = 0$ which means that we have our estimate

$$\gamma_k = \alpha_k + \sum_{n=1}^{N} \phi_{nk} \tag{A.1}$$

Next, we maximize the lower bound with respect to ϕ_n (the variational distribution of z_n) by collecting all terms containing ϕ_{nk}:

$$\mathbb{L}[\phi_{nk}] = \phi_{nk}(\tilde{\psi}_k + \log \mathrm{N}(x_n \mid \mu_k, \sigma^2)) - \phi_{nk} \log \phi_{nk} - \lambda(\sum_k \phi_{nk} - 1)$$

(note using *Lagrange multipliers* to enforce the constraint $\sum_k \phi_{nk} = 1$) and taking derivative with respect to ϕ_{nk},

$$\frac{\partial}{\partial \phi_{nk}} \mathbb{L}[\phi_{nk}] = \tilde{\psi}_k + \log \mathrm{N}(x_n \mid \mu_k, \sigma^2) - \log \phi_{nk} - 1 - \lambda$$

and setting it equal to zero we have

$$\phi_{nk} \propto \exp(\tilde{\psi}_k) \mathrm{N}(x_n \mid \mu_k, \sigma^2) \tag{A.2}$$

$$= \frac{\exp \psi(\gamma_k)}{\exp \psi(\sum_k \gamma_k)} \mathrm{N}(x_n \mid \mu_k, \sigma^2) \tag{A.3}$$

In every E-step of the algorithm we alternate between updating γ by Equation A.1 and ϕ_n by Equation A.3 until convergence.